REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTRY OF HIGHER EDUCATION
AND SCIENTIFIC RESEARCH

HIGHER SCHOOL IN APPLIED SCIENCES
–T L E M C E N–

وزارة التعليم العالي والبحث العلمي

المدرسة العليا في العلوم التطبيقية
- تلمسان -

Mémoire de fin d'étude

Pour l'obtention du diplôme d'Ingéniorat

Filière : Automatique

Spécialité : Automatique

Présenté par : BORSALI Atika

Thème

Conception et implémentation d'un système intelligent de surveillance continue des stocks basés sur la détection visuelle de médicaments

Soutenu publiquement, le 22/06/2026, devant le jury composé composé de :

Dr CHIALI Anisse	MCA	ESSA. Tlemcen	Président
Dr FANDI Wahiba	MCB	ESSA. Tlemcen	Directrice de mémoire
Dr BEKKAOUI Mokhtaria	MCB	ESSA. Tlemcen	Co-Directrice de mémoire
Dr KARAOUZENE Zoheir	MCB	ESSA. Tlemcen	Examinateur 1
Dr MOKHTARI Rida	MCA	ESSA. Tlemcen	Examinateur 2

Année universitaire : 2025/2026


بسم الله الرحمن الرحيم

Dédicaces

Je dédie ce mémoire

à mes chers parents,

à mon frère Nazim,

à toute ma famille.

Remerciements

Il reste difficile de citer tous ceux qui le méritent et j'espère n'oublier personne au moment de la finalisation de mon mémoire de fin d'études.

Je tiens avant tout à remercier Allah, le Tout-Puissant, qui m'a donné la force, la santé et la volonté de mener à bien ce travail.

Je remercie en particulier mes encadrantes Madame FANDI Wahiba et Madame BEKKAOUI Mokhtaria pour leur patience, leur disponibilité, ainsi que leurs judicieux conseils. Comme je voudrais adresser mes sincères remerciements et toute ma gratitude à Madame NEDJAR Imane, qui a analysé mes résultats et m'a prodigué des conseils d'une grande richesse, ainsi qu'à Monsieur M'HAMEDI Mohamed, dont l'aide m'a été précieuse pour identifier et corriger les erreurs liées au matériel.

Je tiens à exprimer ma profonde gratitude à la pharmacie MERZOUK, et tout particulièrement à son propriétaire Monsieur MERZOUK Azeddine, pour son accueil exceptionnel. Il m'a non seulement permis de prendre des photos des médicaments nécessaires à mon travail, mais il m'a également généreusement communiqué leurs données de ventes. Sa disponibilité, sa confiance et sa grande générosité m'ont été d'une aide précieuse. Qu'il trouve ici l'assurance de mes sincères remerciements.

Je remercie également le responsable du FABLAB Monsieur ADJIM Ramz-Eddine Abderazak pour son aide précieuse et son accompagnement tout au long de la réalisation de mon prototype. Il a su mettre à ma disposition les machines et les outils nécessaires, me guider dans les étapes techniques, et me prodiguer de précieux conseils chaque fois que j'étais confronté à une difficulté. Grâce à sa disponibilité et sa patience, j'ai pu concrétiser ce projet. Qu'il trouve ici l'expression de ma profonde gratitude.

Je remercie Monsieur CHIALI Anisse de m'avoir fait l'honneur de présider mon jury de mémoire.

Je remercie également Monsieur KARAOUZENE Zoheir et Monsieur MOKHTARI Rida qui ont bien voulu juger ce travail.

À mes chers parents, aucun mot ne saurait exprimer assez ma gratitude. Vous avez été mon refuge, ma force et mon exemple. Vos sacrifices, votre amour inconditionnel et vos encouragements constants m'ont permis d'arriver là où je suis. Je n'y serais jamais parvenu sans vous. Ce travail est le vôtre, il est l'aboutissement de tout ce que vous m'avez transmis. Merci pour tout, du plus profond de mon cœur. Et pour terminer, mon petit frère : tu es important pour moi.

Table des matières

Introduction générale **12**

1 État de l'art et concepts théoriques **14**

1.1 Introduction 14
1.2 Gestion des stocks pharmaceutiques 14
1.2.1 Définition et importance 14
1.2.2 Méthodes traditionnelles de gestion de stock 15
1.2.3 Défis liés aux stocks pharmaceutiques 16
1.3 Industrie 4.0 et entrepôts intelligents [2] 17
1.3.1 Applications dans le domaine pharmaceutique : Pharma 4.0 [3] 18
1.3.2 Les technologies de l'Industrie 4.0 18
1.4 Vision par ordinateur 19
1.5 Présentation de YOLO 19
1.5.1 Architecture du réseau 20
1.5.2 Avantages de YOLO 20
1.6 Présentation du Raspberry Pi 20
1.6.1 Caractéristiques matérielles 21
1.6.2 Avantages et limites du Raspberry Pi 22
1.6.3 Les différentes versions et le système d'exploitation 22
1.6.4 Les utilisations du Raspberry Pi dans l'industrie 22
1.7 Prévision de la demande 23
1.7.1 Méthodes classiques 23
1.7.2 Approches basées sur le Machine Learning 23
1.7.3 Approches basées sur le Deep Learning [12] 24
1.8 Travaux existants 26
1.9 Conclusion 27

2 Analyse et conception du système **28**

2.1 Introduction 28
2.2 Analyse des besoins 28
2.3 Architecture générale du système 33
2.3.1 Description de la couche matérielle 34

2.3.2 Caméra . . . 36
2.3.3 Description de la couche logicielle . . . 37
2.4 Modélisation UML . . . 39
2.4.1 Diagramme de cas d'utilisation . . . 39
2.5 Scénario de fonctionnement du système PharmaVision 4.0 . . . 47
2.5.1 Capture d'image . . . 47
2.5.2 Détection des produits . . . 47
2.5.3 Mise à jour du stock . . . 48
2.5.4 Stockage dans la base de données . . . 48
2.5.5 Prédiction de la demande . . . 49
2.5.6 Affichage sur le tableau de bord . . . 49
2.6 Conclusion . . . 50

3 Réalisation et résultats expérimentaux 51
3.1 Création du dataset . . . 51
3.1.1 Collecte des images . . . 51
3.1.2 Annotation des données . . . 52
3.2 Entraînement du modèle YOLO . . . 52
3.2.1 Paramètres d'entraînement . . . 52
3.2.2 Évaluation des performances . . . 53
3.2.3 Résultats expérimentaux . . . 53
3.3 Module de prévision de la demande . . . 54
3.3.1 Collecte des données . . . 54
3.3.2 Prétraitement des données . . . 55
3.3.3 Architecture du modèle . . . 56
3.3.4 Entraînement du modèle . . . 57
3.3.5 Résultats et interprétations . . . 58
3.4 Détection en temps réel . . . 58
3.4.1 Déploiement sur Raspberry Pi . . . 58
3.4.2 Paramètres de configuration . . . 59
3.4.3 Chargement du modèle NCNN . . . 59
3.4.4 Initialisation de la caméra . . . 59
3.4.5 Boucle de détection en temps réel . . . 59
3.4.6 Le Serveur Flask et ses Routes . . . 60
3.4.7 Architecture multi-thread . . . 60
3.5 Le Serveur Central . . . 60
3.5.1 La base de données SQLite . . . 61
3.5.2 Les bibliothèques utilisées . . . 61
3.5.3 La boucle de collecte . . . 61

3.5.4 Les routes API REST 62
3.5.5 Architecture multi-thread 62
3.5.6 Séquence de démarrage 63
3.6 Tableau de bord (Dashboard) 63
3.6.1 Structure de l'interface (HTML/CSS/JS) 63
3.6.2 Récupération des données depuis le serveur et les deux Raspberry Pi 65
3.6.3 Fusion des inventaires et gestion des alertes 65
3.6.4 Historique des variations et exports 65
3.7 Discussion générale des résultats 65
3.8 Conclusion 66

Conclusion générale **67**

Annexe A **69**

Table des figures

1.1 Architecture du Bi-LSTM [12]. 26

2.1 Réponse de la question 1 29
2.2 Réponse de la question 2 29
2.3 Réponse de la question 3 30
2.4 Réponse de la question 4 30
2.5 Réponse de la question 5 30
2.6 Réponse de la question 6 31
2.7 Réponse de la question 7 31
2.8 Réponse de la question 8 32
2.9 Réponse de la question 9 32
2.10 Réponse de la question 10 32
2.11 Prototype du projet de fin d'étude. 34
2.12 Raspberry Pi 3 Model B 35
2.13 Caméra Raspberry Pi module v2 36
2.14 Diagramme de cas d'utilisation du système PharmaVision 4.0 39
2.15 Diagramme de séquence de mise à jour de l'inventaire et de gestion des alertes de PharmaVision 4.0 43
2.16 Diagramme de séquence de la consultation des alertes via le tableau de bord 44
2.17 Diagramme de séquence « Consulter le stock » 45
2.18 Diagramme de séquence du processus de détection et de comptage . . . 46
2.19 Diagramme de séquence du processus de prédiction de la demande (modèle Bi-LSTM) 47

3.1 Architecture du modèle BI-LSTM. 56
3.2 Résultat de l'entrainement. 58
3.3 Courbes d'évaluation du modèle Bi-LSTM 58
3.4 Tableau de bord section Accueil 64
3.5 Tableau de bord section Prédiction IA 64

Liste des tableaux

2.1 Bibliothèques logicielles utilisées 38

3.1 Contenu de l'un des fichiers Excel 55
3.2 Architecture du modèle Bi-LSTM 57
3.3 Étapes de la boucle de détection 60
3.4 Bibliothèques Python du serveur central 61
3.5 Routes API REST du serveur central 62

Table des abréviations

Abréviation	Signification
ANN	Artificial Neural Network (Réseau de Neurones Artificiels)
API	Application Programming Interface (Interface de Programmation Applicative)
ARM	Advanced RISC Machine (Architecture de processeur)
BGR	Blue Green Red (format d'image)
Bi-LSTM	Bidirectional Long Short-Term Memory (LSTM Bidirectionnel)
CEC	Constant Error Carousel
COCO	Common Objects in Context (base de données d'images)
CONF	Confidence threshold (Seuil de confiance)
CPU	Central Processing Unit (Unité Centrale de Traitement)
CSI	Camera Serial Interface (Interface Série Caméra)
CSV	Comma-Separated Values (format de fichier tabulaire)
DSI	Display Serial Interface (Interface Série Écran)
ESSA	École Supérieure en Sciences Appliquées
Fast R-CNN	Fast Region-based Convolutional Neural Network
FIFO	First In, First Out (Premier Entré, Premier Sorti)
Flask	Framework web léger pour Python
GPU	Graphics Processing Unit (Unité de Traitement Graphique)
GPIO	General Purpose Input Output (Entrées/Sorties à Usage Général)
HDMI	High-Definition Multimedia Interface
HTTP	HyperText Transfer Protocol
IA	Intelligence Artificielle
IoT	Internet of Things (Internet des Objets)
JAT	Juste-À-Temps
JSON	JavaScript Object Notation (format d'échange de données)
JPEG	Joint Photographic Experts Group (format d'image)
LCD	Liquid Crystal Display (Écran à Cristaux Liquides)
LED	Light Emitting Diode (Diode Électroluminescente)
LIFO	Last In, First Out (Dernier Entré, Premier Sorti)
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error (Erreur Absolue Moyenne)

Abréviation	Signification
mAP	mean Average Precision (Précision Moyenne)
mAP50	mean Average Precision à un seuil IoU de 50%
mAP50-95	mean Average Precision pour des seuils IoU de 50% à 95%
MAPE	Mean Absolute Percentage Error (Erreur Absolue Moyenne en Pourcentage)
MJPEG	Motion JPEG (format vidéo)
ML	Machine Learning (Apprentissage Automatique)
MSE	Mean Squared Error (Erreur Quadratique Moyenne)
NCNN	Neural network inference framework optimisé pour CPU ARM
NMS	Non-Maximum Suppression
PC	Personal Computer (Ordinateur Personnel)
PFE	Projet de Fin d'Études
QEC	Quantité Économique à Commander
RAM	Random Access Memory (Mémoire Vive)
RFID	Radio Frequency Identification (Identification par Radiofréquence)
RMSE	Root Mean Square Error (Racine de l'Erreur Quadratique Moyenne)
R^2	Coefficient de Détermination
REST	Representational State Transfer
SD	Secure Digital (carte mémoire)
SEQ_LEN	Sequence Length (Longueur de Séquence)
SoC	System on Chip (Système sur Puce)
SQLite	Structured Query Language Lite (base de données légère)
TXT	Text file (format de fichier texte)
USB	Universal Serial Bus
VRAM	Video Random Access Memory (Mémoire Vidéo Vive)
Wi-Fi	Wireless Fidelity (réseau sans fil)
XLS	Excel Spreadsheet (format de feuille de calcul Microsoft Excel)
YOLO	You Only Look Once (algorithme de détection d'objets)
YOLOv8	You Only Look Once – version 8

Introduction générale

La gestion des stocks en pharmacie est nettement plus critique que dans un commerce ordinaire. Une rupture de stock peut mettre en danger la vie d'un patient. Chaque produit doit être tracé et comptabilisé sous peine de ne pas respecter des obligations légales très strictes. En plus de cela, la demande est imprévisible et parfois urgente. En pharmacie, une erreur de stock n'est pas une simple perte financière mais c'est d'abord un risque humain. Aujourd'hui, la plupart des pharmacies utilisent des systèmes de codes-barres ou QR codes couplés à un logiciel de gestion. Chaque médicament est scanné à l'entrée et à la sortie du stock, ce qui permet un suivi automatisé des mouvements. Cette méthode a largement remplacé les inventaires purement manuels et représente une avancée réelle. Cependant, ce système est loin d'être parfait. Il a plusieurs gros défauts. La première limite est sa dépendance totale à l'action humaine. La fiabilité du système repose entièrement sur la réalisation systématique de chaque scan. Or, en période de très grande activité ou en situation d'urgence, il arrive que certains médicaments soient retirés sans être scannés, par oubli ou par manipulation rapide, ce qui crée immédiatement un écart entre le stock réel et celui affiché dans le logiciel. Deuxièmement, le système n'a pas accès à la réalité physique du stock. Il enregistre des transactions, sans jamais voir ce qui se passe vraiment dans les rayons. Ainsi, un produit mal rangé, volé ou endommagé reste invisible pour le logiciel, qui continuera à l'indiquer comme étant disponible. Enfin, les systèmes de codes-barres classiques proposent parfois des alertes de stock minimum, mais ces seuils sont définis manuellement par le pharmacien et restent fixes. Ils ne suivent pas l'évolution réelle de la demande, par exemple lors d'une épidémie saisonnière ou d'une hausse soudaine des prescriptions. Cela les rend incapables d'anticiper intelligemment une rupture de stock avant qu'elle n'arrive. Un questionnaire mené auprès de pharmaciens en Algérie montre que tous les répondants sont confrontés à des ruptures de stock. Pour 57,1 Ce que ces chiffres montrent, c'est que le problème n'est pas un manque de sérieux de la part des pharmaciens, mais c'est plutôt un manque d'outils adaptés à la réalité du terrain. Et c'est ce que la technologie d'aujourd'hui essaye de combler. C'est exactement dans ce contexte que se situe ce travail. S'appuyant sur les avancées de l'Industrie 4.0 et du domaine de la Pharma 4.0, il propose une solution intelligente et embarquée pour la supervision automatique des stocks pharmaceutiques, nommée PharmaVision 4.0. Cette solution repose sur trois piliers technologiques complémentaires : la vision par ordinateur qui observe et compte les médicaments en temps réel, le Deep Learning qui anticipe la demande future, et une architecture distribuée pour centraliser, archiver et visualiser toutes ces informations au travers d'un tableau de bord accessible au

pharmacien. Concrètement, des caméras reliées à des cartes Raspberry Pi surveillent en permanence les étagères de la pharmacie. Chaque image capturée est analysée par un modèle YOLOv8, qui a été entraîné à reconnaître 105 médicaments réels, collectés à la Pharmacie MERZOUK de Tlemcen. Parallèlement, un modèle Bi-LSTM exploite 41 mois d'historique de ventes pour prédire les demandes futures, afin d'anticiper les ruptures de stock avant qu'elles ne surviennent. L'ensemble du système est piloté par un serveur central qui rassemble les données provenant des deux Raspberry Pi et les affiche sur un tableau de bord web interactif. Ce tableau permet au pharmacien de consulter l'état des stocks, de recevoir des alertes automatiques, de suivre l'historique des mouvements et d'exporter les données. Ce mémoire est structuré en trois chapitres. Le premier pose les bases théoriques et technologiques du projet. Il passe en revue les méthodes classiques de gestion des stocks, les principes de l'Industrie 4.0, la vision par ordinateur, le modèle YOLO, la plateforme Raspberry Pi, ainsi que les différentes approches de prévision de la demande, depuis les méthodes statistiques traditionnelles jusqu'aux réseaux de neurones profonds comme LSTM et Bi-LSTM. Le deuxième chapitre est consacré à l'analyse des besoins réels du terrain, à la présentation de l'architecture générale du système PharmaVision 4.0 et à sa modélisation sous forme de diagrammes UML. Enfin, le troisième chapitre décrit la mise en œuvre concrète du système, depuis la construction des datasets jusqu'à l'entraînement des modèles, ainsi que le déploiement sur Raspberry Pi et l'analyse des résultats expérimentaux obtenus.

Chapitre 1

État de l'art et concepts théoriques

1.1 Introduction

La gestion intelligente des stocks pharmaceutiques est une nécessité pour les pharmacies, les hôpitaux et les cliniques. Avec ce nombre énorme de produits à suivre, les règles de traçabilité de plus en plus strictes, et le risque de rupture ou de péremption, les méthodes traditionnelles ne suffisent plus pour gérer le stock. C'est dans ce contexte que l'Industrie 4.0 a offert de nouvelles solutions et de nouvelles technologies qui combinent à la fois l'automatisation, la communication et l'intelligence artificielle pour améliorer l'efficacité et la fiabilité de la gestion des stocks.

Dans ce chapitre, on pose les bases théoriques et technologiques de notre projet en définissant d'abord la gestion des stocks pharmaceutiques, ses méthodes classiques et les défis qu'elle engendre quotidiennement. Ensuite, on parle des principes de l'Industrie 4.0 et leur application dans le domaine pharmaceutique à travers Pharma 4.0. Nous abordons la vision par ordinateur, qui est le cœur de notre système, puis nous présentons le modèle de détection d'objets en temps réel utilisé dans ce projet, qui est YOLO. Nous décrivons également le Raspberry Pi, la plateforme matérielle embarquée choisie pour ce système. Après cela, nous présentons les différentes approches de prévision de la demande, en passant des méthodes classiques jusqu'aux modèles de Deep Learning tels que le LSTM et le Bi-LSTM. Enfin, nous présentons quelques travaux déjà existants et nous terminons par une conclusion.

1.2 Gestion des stocks pharmaceutiques

1.2.1 Définition et importance

Dans tout établissement de santé, qu'il s'agisse d'un hôpital ou d'une clinique, la gestion des stocks représente une activité centrale et très nécessaire. La gestion de stock pharmaceutique s'organise en trois grandes étapes :

— **L'achat :** c'est l'étape où on évalue les besoins, c'est-à-dire qu'on estime quels produits seront nécessaires et en quelle quantité.

— **La conservation :** une fois les produits reçus, ils doivent être stockés dans de bonnes conditions, car certains médicaments se dégradent s'ils sont exposés à une chaleur excessive ou à la lumière.

— **La distribution :** il s'agit ensuite de fournir les produits aux services qui en ont besoin, et surtout au bon moment et en bonne quantité.

Une mauvaise gestion des stocks peut avoir des conséquences graves. Si un médicament est en rupture, le traitement du patient peut être retardé, ce qui affectera directement sa santé. Une pharmacie a la responsabilité de fournir des médicaments conformes et de qualité, non périmés ou endommagés. La traçabilité d'un produit pharmaceutique est très importante : savoir quel médicament a été vendu, à quel patient, à quelle date, est indispensable en cas de problème ou de contrôle. Acheter trop de produits est du gaspillage, et acheter trop peu peut causer une pénurie. Pour trouver le bon équilibre, il faut assurer d'abord une bonne gestion de stock [1].

1.2.2 Méthodes traditionnelles de gestion de stock

Il existe plusieurs méthodes traditionnelles de gestion de stock, parmi lesquelles :

1. **Le point de commande :** C'est un niveau de stock prédéfini à partir duquel on déclenche automatiquement une nouvelle commande, c'est-à-dire qu'on commande avant que les rayons soient vides. Il est calculé en tenant compte du délai de livraison du fournisseur et de la vitesse à laquelle les produits sont vendus. Le point de commande est souvent accompagné d'un stock de sécurité, qui est une petite réserve de produits gardée de côté pour faire face aux imprévus comme les retards de livraison ou les pics soudains de la demande.
2. **L'analyse ABC :** Tous les produits n'ont pas la même valeur ni la même importance. L'analyse ABC permet justement de les classer en trois groupes :
   — **La catégorie A** regroupe les produits les plus importants représentant environ 80% des revenus mais seulement 20% des produits.
   — **La catégorie B** correspond aux produits de moyenne importance générant environ 15% des revenus.
   — **La catégorie C** regroupe les produits peu demandés, ne représentant que 5% des revenus, parfois appelés stock mort car ils restent en entrepôt sans bouger.
3. **La Quantité Économique à Commander (QEC) :** Connue sous le nom de formule de Wilson, elle répond à la question suivante : combien faut-il commander à chaque fois pour que cela coûte le moins cher possible ? En effet, commander trop souvent engendre des frais administratifs et de transport répétés, mais commander en grande quantité augmente les coûts de stockage. La QEC cherche donc le juste milieu entre ces deux extrêmes.
4. **Le Juste-à-Temps (JAT) :** Popularisée par le constructeur automobile Toyota, c'est une méthode qui a pour but de synchroniser la production à la demande

des clients. Autrement dit, on ne stocke que ce dont on a besoin, au moment où on en a besoin. L'objectif est de réduire au maximum les coûts de stockage. C'est une approche efficace, mais l'entreprise devient très dépendante de ses fournisseurs, car les produits doivent être livrés directement après la production, ce qui nécessite des fournisseurs ponctuels et des relations solides avec eux.

5. **La méthode FIFO (First In, First Out) :** Son principe est le suivant : la première unité entrée en stock est celle qui sort en premier, ce qui signifie que le lot le plus ancien est toujours utilisé en priorité. Cette méthode est très simple et particulièrement adaptée aux produits périssables, dont une conservation prolongée peut entraîner une diminution de valeur ou de qualité, comme les produits agroalimentaires, pharmaceutiques ou cosmétiques.
6. **La méthode LIFO (Last In, First Out) :** Le LIFO fonctionne à l'inverse du FIFO : la dernière unité entrée dans le stock est celle qui en sort en priorité. Les articles les plus anciens restent donc en stock plus longtemps. Cette approche est souvent utilisée dans les situations où les prix des produits ont tendance à augmenter avec le temps. En revanche, cette méthode est totalement déconseillée pour les produits périssables.
7. **La méthode FEFO (First Expired, First Out)** : La méthode FEFO est similaire à celui du FIFO, elle consiste à prélever en premier les produits dont leur date limite de consommation est la plus proche. Cette stratégie est indispensable dans les industries traitant des produits périssables ce qui permet de diminuer les risques de perte et elle entraîne une meilleure rotation des stocks.

1.2.3 Défis liés aux stocks pharmaceutiques

Voici quelques obstacles que rencontrent les pharmaciens au quotidien :

— Trouver le bon équilibre entre commander trop peu, ce qui risque de priver un patient d'un médicament urgent, et commander trop, ce qui immobilise de l'argent dans des produits stockés.

— Chaque médicament a une date de péremption. S'il n'est pas repéré à temps et retiré du stock, il devient une perte pure. Ce problème est beaucoup plus complexe quand le stock est important.

— Tous les médicaments ne supportent pas les mêmes conditions : certains doivent être conservés au froid, d'autres à l'abri de la lumière ou dans un endroit sec, ce qui complique leur gestion.

— Il y aura toujours des produits qui restent sur les étagères sans se vendre, représentant une perte financière pour la pharmacie.

— Un produit mal placé ou mal enregistré dans le système fait que le stock affiché

ne reflète plus la réalité, ce qui peut entraîner de mauvaises commandes ou des produits introuvables au moment où on en a besoin.

— Parfois, le problème ne vient pas de la pharmacie elle-même, mais des fabricants ou des grossistes qui ne peuvent plus fournir certains médicaments, ce qui met la pharmacie dans l'impossibilité de répondre à la demande.

1.3 Industrie 4.0 et entrepôts intelligents [2]

L'Industrie 4.0 consiste à connecter des machines, des logiciels et des personnes pour qu'ils travaillent ensemble de manière intelligente, en échangeant des informations en temps réel et en prenant certaines décisions sans attendre qu'un humain intervienne. Elle ne modernise pas seulement les usines, mais touche aussi les entrepôts, les pharmacies, les hôpitaux, les transports et plusieurs autres domaines impliquant la gestion des produits, des flux et des informations.

Les principaux objectifs de l'Industrie 4.0 sont :

— Automatiser le maximum de tâches afin de réduire les dépenses inutiles et les pertes dues aux erreurs humaines.

— Gagner du temps sur les tâches répétitives.

— Mieux anticiper les besoins pour éviter les mauvaises surprises.

— Donner une vision claire et en temps réel de ce qui se passe.

L'Industrie 4.0 repose sur cinq piliers :

— **Communication entre les systèmes :** Dans beaucoup d'organisations, les différents outils informatiques ne communiquent pas entre eux, ce qui cause des pertes de temps, des erreurs et un manque de visibilité sur la situation globale. Ce premier pilier répond exactement à ce problème en faisant en sorte que tous ces outils se connectent et s'échangent des données automatiquement.

— **Les entrepôts intelligents :** C'est un espace de stockage dans lequel on trouve des capteurs, des robots et des logiciels qui effectuent la plupart des tâches.

— **Transformation des données en décisions :** Les données générées chaque jour par l'entrepôt sont analysées grâce au Big Data, un outil qui transforme des millions d'informations brutes en décisions simples et utiles : quoi commander, quand et en quelle quantité.

— **Des objets connectés au service du stock :** Gérer un stock, c'est savoir à tout moment où se trouve chaque produit, dans quelles conditions il est conservé, et quand il faut le réapprovisionner. Les objets connectés comme l'IoT et le RFID rendent tout cela possible et automatique, avec beaucoup moins d'intervention humaine.

— **Des machines qui s'adaptent à leur environnement :** Les systèmes cyber-physiques sont des machines capables de percevoir ce qui se passe autour d'elles et d'ajuster leur comportement pour s'adapter aux différents changements de leur environnement.

1.3.1 Applications dans le domaine pharmaceutique : Pharma 4.0 [3]

L'Industrie 4.0 a transformé beaucoup de secteurs, mais le domaine pharmaceutique est un cas particulier et très strict, car une erreur peut coûter une vie. C'est pourquoi est née la Pharma 4.0, une version adaptée aux exigences du médicament. La transition vers la Pharma 4.0 repose sur quatre changements fondamentaux :

— **Numériser le tout :** elle propose de remplacer les fiches papier et registres manuscrits par des données fiables et accessibles à tout moment.

— **Faire circuler l'information :** chaque service ne doit plus garder ses données pour lui, mais les partager avec les autres services.

— **Collaborer avec les autorités :** prouver que les produits respectent les normes devient beaucoup plus simple et rapide grâce aux outils numériques.

— **Gérer les risques intelligemment :** plutôt que de contrôler chaque étape de façon isolée, on regarde l'ensemble du processus pour identifier ce qui est vraiment critique, ce qui permet de se concentrer là où cela compte.

1.3.2 Les technologies de l'Industrie 4.0

Il existe plusieurs outils technologiques qui permettent de faire fonctionner une usine de façon intelligente et automatique, parmi lesquels :

— **Les automates programmables :** ce sont des ordinateurs industriels très robustes qui contrôlent les machines et les équipements.

— **L'intelligence artificielle et le Machine Learning :** l'apprentissage automatique donne à un système informatique la capacité d'apprendre à partir des données qu'on lui fournit et de prendre des décisions dans des cas qu'il n'a jamais rencontrés.

— **La maintenance prédictive :** les machines sont équipées de capteurs qui surveillent en permanence leur état (vibrations, température, usure des pièces) et dès qu'un indicateur dépasse un seuil prédéfini, une alerte se déclenche pour intervenir immédiatement. C'est un gain de temps et d'argent, surtout quand un arrêt de production peut avoir des conséquences importantes.

1.4 Vision par ordinateur

La vision par ordinateur est un domaine de l'intelligence artificielle qui vise à permettre aux machines de voir, analyser et interpréter le contenu d'une image ou d'une vidéo de la même façon que l'œil humain. En d'autres termes, c'est l'ensemble des techniques qui permettent à un ordinateur d'extraire des caractéristiques utiles à partir d'images numériques, de les comprendre et ensuite de prendre des décisions.

La vision par ordinateur recouvre plusieurs tâches, comme :

— **Classification d'images :** reconnaître un objet dans une image (un chat, une voiture, etc.).

— **Détection d'objets :** localiser et identifier un ou plusieurs objets dans une seule image.

— **Segmentation :** délimiter avec une grande précision les contours de chaque objet.

— **Reconnaissance d'actions humaines :** comprendre ce que fait une personne sur une vidéo.

La vision par ordinateur s'appuie sur le traitement d'image, les mathématiques (algèbre linéaire, statistiques, calcul différentiel) et surtout l'apprentissage automatique (Machine Learning), en particulier les réseaux de neurones profonds (Deep Learning), qui ont révolutionné le domaine en permettant aux machines d'apprendre automatiquement les caractéristiques visuelles à partir de grandes quantités de données. Ce domaine scientifique existe depuis les années 1960, et il est aujourd'hui présent dans plusieurs secteurs tels que l'industrie, la robotique, l'agriculture, la médecine et le militaire.

1.5 Présentation de YOLO

Avant l'apparition de YOLO, détecter des objets dans une image était un processus long et compliqué : les systèmes faisaient la détection en deux étapes principales. D'abord, ils découpaient l'image en morceaux et les analysaient un par un, puis ils essayaient de combiner les résultats. Cette méthode était efficace mais très lente.

YOLO a réduit ces étapes en regardant l'image une seule fois et en produisant immédiatement les résultats. Au lieu de découper le travail en plusieurs étapes, il utilise un seul réseau de neurones qui analyse l'image entière et produit non seulement la classe de l'objet, mais aussi sa localisation précise dans l'image, et tout cela en quelques millisecondes.

Voici les étapes de fonctionnement de YOLO :

1. D'abord, il divise l'image en une grille de cases, où chacune est responsable de détecter les objets dans sa zone.

2. Ensuite, pour chaque case, le réseau dessine des rectangles autour des objets détectés, appelés boîtes englobantes, accompagnés d'un score de confiance indiquant à quel point le modèle est sûr de la présence d'un objet.
3. Enfin, il calcule la probabilité de classe, qui traduit la chance que l'objet détecté appartienne à une catégorie précise.

En combinant le score de confiance et la probabilité de classe, YOLO obtient un score final pour chaque objet détecté.

1.5.1 Architecture du réseau

Le réseau de neurones de YOLO est composé de 24 couches convolutives qui cherchent et extraient les caractéristiques importantes comme les contours, les formes et les textures. Ces couches sont suivies de 2 couches entièrement connectées qui prennent toutes ces informations et les transforment en prédictions finales sur la position et la classe des objets. Il existe beaucoup de versions des modèles YOLO, et chacune apporte une meilleure architecture du réseau de neurones qui lui permet de reconnaître plus d'objets, d'améliorer la précision et la vitesse.

1.5.2 Avantages de YOLO

YOLO présente trois avantages par rapport aux autres modèles de Deep Learning :

— Sa rapidité, en ne passant qu'une seule fois dans le réseau pour tout détecter.

— Il voit l'image entière et non de petits morceaux, ce qui lui permet de comprendre le contexte global du contenu, et il fait en plus trois fois moins d'erreurs que le Fast R-CNN.

— YOLO est capable de fonctionner correctement même dans des situations qu'il n'a jamais rencontrées pendant son entraînement, contrairement à ses concurrents.

Sur la base de données d'images PASCAL VOC, YOLO arrive à détecter les objets avec $63,4\%$ de précision à 45 images par seconde. La version Fast YOLO atteint $52,7\%$ à 155 images par seconde. Lorsqu'il est associé à Fast R-CNN, le score monte à 75%. Grâce à sa vitesse et sa robustesse, YOLO est aujourd'hui appliqué dans de nombreux domaines tels que la conduite autonome, les systèmes d'aide aux personnes malvoyantes, la robotique ou encore la surveillance vidéo intelligente [4].

1.6 Présentation du Raspberry Pi

Eben Upton et son équipe ont eu l'idée de créer un ordinateur tellement petit et abordable que n'importe qui peut l'avoir et apprendre à programmer sans risquer d'abîmer un ordinateur coûteux. Après plusieurs années de développement et de prototypes,

la fondation Raspberry Pi a été créée officiellement, et le premier Raspberry Pi a été mis en vente en février 2012 à un prix raisonnable comparé à un ordinateur classique. En seulement deux ans, 2 millions d'unités avaient été vendues.

Le Raspberry Pi est un ordinateur complet de taille très réduite. Il possède les mêmes composants qu'un ordinateur — un processeur, une mémoire et des ports de connexion — mais dans un format minuscule. Un microcontrôleur simple comme l'Arduino ne fait qu'exécuter un programme informatique fixe, mais un Raspberry Pi fait tourner un vrai système d'exploitation et peut gérer plusieurs tâches en même temps. De plus, on peut le connecter à des périphériques externes comme un clavier, une souris et un écran, et l'utiliser presque comme un PC classique.

1.6.1 Caractéristiques matérielles

Plus précisément, un Raspberry Pi se compose de :

— **Un processeur (CPU) :** c'est le cerveau de l'appareil. Il calcule et exécute toutes les instructions. Sur le Raspberry Pi, il est de type ARM, une architecture utilisée dans les smartphones pour sa consommation réduite d'énergie tout en étant suffisamment puissante.

— **Un SoC (System on Chip) :** c'est un composant qui regroupe sur une seule petite pièce le processeur, le processeur graphique (GPU) et d'autres circuits, permettant de réduire la taille et le coût de la carte.

— **Une mémoire RAM :** elle stocke temporairement les données pendant que le système est en marche.

— **Une carte SD :** le Raspberry Pi n'a pas de disque dur ; on lui installe une carte SD pour stocker le système d'exploitation et les fichiers.

— **Des ports USB et Ethernet :** les ports USB permettent de brancher un clavier, une souris ou une clé Wi-Fi. Le port Ethernet permet de connecter l'appareil à internet via un câble réseau.

— **Un port HDMI :** pour relier le Raspberry Pi à un écran. Il supporte une résolution allant jusqu'à 1920×1200 pixels.

— **Des broches GPIO (General Purpose Input Output) :** ce sont les broches d'entrée/sortie polyvalentes. Il en possède 40, dont 26 utilisables directement pour connecter des capteurs, des LEDs, des moteurs, des boutons, etc.

— **Un connecteur caméra (CSI) :** un port spécial pour brancher un module caméra directement à la carte.

— **Un connecteur écran (DSI) :** un port pour brancher un petit écran LCD via un câble plat.

1.6.2 Avantages et limites du Raspberry Pi

Le premier avantage est évidemment son prix très bas pour sa capacité à faire tourner des programmes, se connecter à internet et interagir avec des composants électroniques. Il fonctionne sous Linux, un système d'exploitation gratuit et libre, ce qui réduit encore plus les coûts. Il consomme également très peu d'électricité, ce qui est très avantageux pour les projets qui tournent en continu.

Malgré ses nombreux avantages, cet appareil présente quelques limitations importantes à prendre en compte. Il ne peut pas remplacer un PC complètement, car il possède peu de RAM et de puissance pour les tâches lourdes. Il n'a pas non plus d'horloge interne avec batterie, ce qui signifie qu'il perd l'heure à chaque coupure de courant si on ne le connecte pas à internet pour la synchroniser.

1.6.3 Les différentes versions et le système d'exploitation

Au fil des années, plusieurs modèles ont été lancés, chacun apportant des améliorations par rapport au précédent. Les principales différences entre eux sont leur prix, leur taille, leur puissance de calcul, leur quantité de RAM et leurs ports de connexion, ce qui permis à chacun de s'adapter à un type de projet spécifique.

Comme n'importe quel ordinateur, le Raspberry Pi a besoin d'un système d'exploitation pour fonctionner et exécuter des programmes. Le système officiel s'appelle Raspbian, basé sur Debian (une version populaire de Linux) ; il est gratuit et optimisé pour le Raspberry Pi. D'autres systèmes sont également compatibles selon les besoins [5].

1.6.4 Les utilisations du Raspberry Pi dans l'industrie

Le Raspberry Pi n'est pas uniquement adapté aux apprentis et aux petits projets, il est aussi utilisé dans l'industrie. Voici ses principales utilisations :

— Le Raspberry Pi peut connecter des machines anciennes au réseau internet, ce qui permet de surveiller à distance des informations importantes comme la chaleur dégagée par une machine ou sa consommation électrique.

— En branchant une caméra sur le Raspberry Pi et en lui ajoutant un programme d'intelligence artificielle, il devient capable d'examiner les produits dans la chaîne de production et de détecter automatiquement les défauts de fabrication.

— Dans le domaine de la gestion des stocks, le Raspberry Pi peut compter automatiquement les produits stockés, suivre leurs déplacements et envoyer des alertes quand un article est en rupture de stock.

— Grâce à des capteurs mesurant le courant et la tension électrique, le Raspberry Pi enregistre combien d'énergie consomme chaque machine, ce qui aide les entreprises

à repérer les équipements les plus énergivores.

— Le Raspberry Pi est aussi utilisé pour surveiller en permanence la température, le taux d'humidité ou la qualité de l'air, et si une anomalie est détectée, une alerte est immédiatement envoyée pour protéger les travailleurs.

— Le Raspberry Pi peut être utilisé pour commander des robots ou des bras mécaniques dans une usine.

— Les entreprises peuvent utiliser le Raspberry Pi pour créer des tableaux de bord sur mesure, adaptés exactement à leurs besoins.

Le Raspberry Pi s'est imposé comme un choix naturel pour les systèmes embarqués. Son prix abordable, sa taille réduite et sa faible consommation le rendent idéal pour des projets concrets, que ce soit un robot, un drone ou tout autre type de projet [6].

1.7 Prévision de la demande

1.7.1 Méthodes classiques

— **Moyenne mobile :** C'est une méthode simple qui prédit la demande future en calculant la moyenne des données passées sur un intervalle de temps fixe. Par exemple, pour prévoir la demande du mois prochain, on calcule la moyenne des 3 ou 6 derniers mois. Son inconvénient est qu'elle accorde le même poids à toutes les données passées dans cet intervalle, sans privilégier les données les plus récentes.

— **Lissage exponentiel :** C'est une méthode plus sophistiquée qui corrige ce défaut des moyennes mobiles, car elle accorde un poids décroissant aux données passées, ce qui signifie que les données les plus récentes ont plus d'influence sur la prévision [7].

— **Régression linéaire :** La régression est une technique qui permet à une machine d'apprendre à faire des prédictions. En lui fournissant des données passées, la machine apprend à prédire des valeurs numériques dans des situations nouvelles. Plus précisément, elle apprend à trouver une relation mathématique qui relie les données en entrée avec la donnée de sortie qu'on souhaite prédire. Ce modèle est largement utilisé dans plusieurs domaines afin de prédire la demande d'un produit, le chiffre d'affaires, la température ou la quantité de pluie. La régression possède à son tour plusieurs types, notamment la régression linéaire, logistique, polynomiale, etc [8].

1.7.2 Approches basées sur le Machine Learning

— **Random Forest :** C'est une technique très populaire dans le domaine de l'exploration de données, où l'on a besoin d'analyser de grandes quantités de données

afin d'en extraire les plus utiles. Random Forest est aussi utilisé pour la classification et la régression. Elle est facile à utiliser et donne des résultats rapidement même pour les problèmes complexes, et peut aussi gérer les données manquantes. Random Forest prend aléatoirement une portion des données d'entraînement (les données passées) et une portion des variables à chaque fois qu'il forme un arbre de décision [9].

— **XGBoost :** C'est un algorithme de Machine Learning qui fonctionne en construisant plusieurs arbres de décision les uns après les autres, où chaque arbre apprend des erreurs commises par le précédent pour améliorer progressivement la prédiction. Grâce à cette approche, il arrive à produire des résultats très précis et dans un temps réduit, même lorsque la quantité de données est grande [10].

— **Réseaux de neurones artificiels (ANN) :** Les réseaux de neurones artificiels sont des modèles inspirés des réseaux de neurones du cerveau humain, où tous les neurones sont interconnectés entre eux, organisés en couches et chaque connexion est caractérisée par une variable appelée poids qui définit la force de la liaison. Ces poids sont ajustés automatiquement lors de l'apprentissage pour améliorer la précision et diminuer les erreurs [11].

1.7.3 Approches basées sur le Deep Learning [12]

LSTM (Long Short-Term Memory)

Avant le LSTM, les réseaux de neurones récurrents apprenaient en comparant leurs prédictions avec les bonnes réponses, calculaient l'erreur et l'utilisaient pour ajuster leurs poids. Le problème de cette méthode est que l'erreur diminue trop vite lorsqu'on remonte en arrière dans la séquence. Ainsi, si une information importante s'était produite 50 ou 100 étapes plus tôt, le réseau ne pouvait pas en tenir compte. Ce problème est appelé le problème des dépendances à long terme, et le LSTM a précisément été créé pour le résoudre. Au lieu de laisser l'erreur disparaître, le LSTM la maintient stable grâce à un mécanisme spécial qui lui permet d'enregistrer les informations importantes même si elles se sont produites très loin dans le passé.

Architecture du LSTM : Le LSTM repose sur trois éléments fondamentaux :

— **La cellule mémoire :** c'est le cœur du système. Elle contient un mécanisme appelé Constant Error Carousel (CEC), une unité connectée à elle-même avec un poids fixe de $1,0$, qui garantit que l'erreur circulant à l'intérieur de la cellule ne change pas au fil du temps, permettant à la cellule de stocker une information de manière stable sur une longue période.

— **La porte d'entrée :** c'est comme un filtre qui protège la cellule contre les informations inutiles qui pourraient perturber et fausser ce qui est déjà stocké.

— **La porte de sortie :** elle contrôle ce qui sort de la cellule, c'est-à-dire à quel moment l'information stockée doit être transmise au reste du réseau, pour éviter de perturber les autres unités avec des informations qui ne sont pas encore utiles.

Ces trois éléments travaillent ensemble dans un même bloc où plusieurs cellules partagent les mêmes portes, ce qui permet de stocker des informations plus complexes. Le réseau est organisé en trois couches : une couche d'entrée, une couche cachée qui contient les cellules mémoire, et une couche de sortie.

Bi-LSTM (Bidirectional LSTM)

Le Bi-LSTM est une version améliorée du LSTM classique. Contrairement au LSTM qui traite une séquence uniquement dans un sens (du début vers la fin), le Bi-LSTM la lit dans les deux sens en même temps, c'est-à-dire du début vers la fin et de la fin vers le début. Ainsi, pour chaque élément de la séquence, le réseau tient compte à la fois de ce qui le précède et de ce qui le suit, ce qui lui permet de mieux comprendre le sens global de la séquence.

Architecture du Bi-LSTM : Le Bi-LSTM est composé de six couches qui travaillent les unes après les autres :

— **La couche d'entrée :** le texte n'est pas transmis en une seule fois ; cette couche est chargée de le découper en mots individuels présentés au réseau un par un en respectant l'ordre.

— **La couche d'embedding :** elle transforme chaque mot en un vecteur numérique qui capture son sens, car un réseau de neurones ne peut pas travailler directement avec des mots.

— **La couche LSTM :** elle traite la séquence de vecteurs dans les deux directions simultanément : un premier LSTM lit du premier au dernier élément, et un deuxième lit dans le sens inverse.

— **La couche entièrement connectée :** elle combine les informations extraites par le LSTM pour préparer la prédiction finale.

— **La couche Softmax :** elle convertit les résultats en probabilités où chaque probabilité correspond à une classe possible, et leur somme est égale à un.

— **La couche de classification :** elle sélectionne la classe ayant la probabilité la plus élevée et la retourne comme résultat final du modèle.

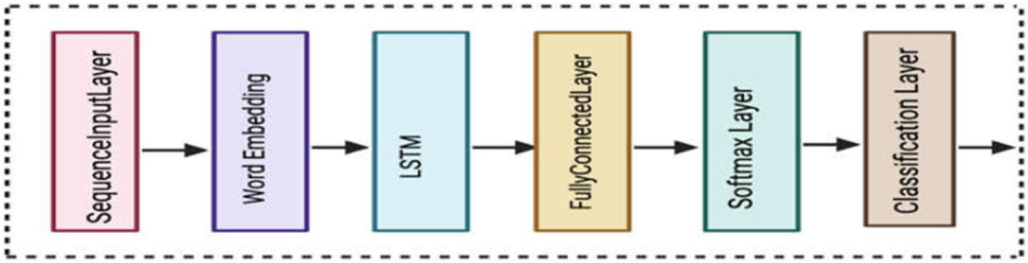


FIGURE 1.1 – Architecture du Bi-LSTM [12].

Les portes du LSTM au sein du Bi-LSTM : Le LSTM dispose de quatre portes qui fonctionnent comme des filtres intelligents :

- **La porte d'oubli :** elle décide quelles informations de l'étape précédente ne sont plus utiles et doivent être effacées. C'est elle qui empêche la cellule de se saturer avec des informations inutiles.
- **La porte d'entrée :** elle décide quelles nouvelles informations de l'étape actuelle méritent d'être stockées dans la cellule mémoire.
- **La porte candidate :** elle génère un ensemble de nouvelles valeurs qui pourraient être ajoutées à la mémoire, en travaillant avec la porte d'entrée pour décider ce qui est vraiment important.
- **La porte de sortie :** elle décide quelles informations stockées dans la cellule doivent être transmises à l'étape suivante et à la couche de sortie.

Ces quatre portes travaillent ensemble à chaque pas de temps pour décider quelles informations retenir ou supprimer, et lesquelles transmettre, ce qui permet au Bi-LSTM de garder en mémoire des liens entre des éléments éloignés dans une séquence [13].

1.8 Travaux existants

- **Dynle (2023) :** ont proposé un projet intitulé *Raspberry Pi Inventory Management System* en 2023, qui est un système capable d'enregistrer automatiquement les opérations d'achat. Il utilise trois cartes Raspberry Pi : la première est branchée avec un module caméra pour capturer des images, la deuxième gère la base de données, et la troisième offre une interface tactile pour l'affichage des résultats. Un serveur Django centralise le traitement des données. Deux modèles ont été intégrés dans ce projet : le YOLOv8 pour la détection des produits achetés, et un modèle de reconnaissance faciale pour identifier le client. Ces données sont sauvegardées dans une base SQLite3, et un chatbot Slack envoie à chaque utilisateur un récapitulatif de ses achats mensuels [14].
- **Wu et Lu (2022) :** Dans le domaine de la gestion automatisée des stocks, ils ont

proposé une méthode de détection en temps réel basée sur YOLOv3. Une caméra fixée au plafond capture des images d'entrepôt et les transmet à un ordinateur industriel pour exécuter le modèle de détection. Les auteurs démontrent que le modèle YOLOv3 atteint un mAP de $96,82\%$ avec un taux d'erreur moyen de seulement $1,61\%$, surpassant YOLOv4 qui n'atteignait que $96,06\%$ [15].

— **Ni, Peng et Liu (2022) :** ont étudié la prévision de la demande logistique pour les produits frais et périssables. Ce type de produit rend la gestion des stocks particulièrement délicate, c'est pourquoi ils proposent un modèle basé sur le Bi-LSTM, entraîné sur une base de données contenant les historiques de ventes, les températures ainsi que les conditions météorologiques. En comparant trois modèles (BP, LSTM et Bi-LSTM) sur cette même base de données, les résultats montrent que le Bi-LSTM est le plus performant, avec un MSE de $0,063$ et un R^2 de $0,858$ [16].

— **Siami-Namini, Tavakoli et Siami Namin (2019) :** ont mené une étude comparative entre trois modèles de prévision de séries temporelles : ARIMA, LSTM et BiLSTM en utilisant des données historiques de vente. Leurs résultats ont montré que le BiLSTM est celui qui a donné le meilleur résultat, dépassant le LSTM classique avec une réduction moyenne de l'erreur de $37,78\%$. Cela s'explique par le fait que le BiLSTM analyse les données dans les deux sens, contrairement au LSTM, ce qui lui permet de mieux saisir les variations passées et futures des données [17].

Contrairement aux travaux existants qui traitent séparément la détection ou la prévision, notre système PharmaVision 4.0 combine les deux dans une architecture embarquée sur Raspberry Pi, avec des données réelles collectées en pharmacie.

1.9 Conclusion

Dans ce chapitre nous avons défini un état de l'art de la gestion des stocks pharmaceutiques, ses méthodes classiques et les défis qu'elle rencontre. Ensuite nous avons vu les principes de l'Industrie 4.0 et leur application dans le domaine pharmaceutique à travers la Pharma 4.0. Nous avons défini la vision par ordinateur, le modèle YOLO, la plateforme Raspberry Pi et les approches de prévision de la demande passant par des méthodes classiques jusqu'aux modèles de Deep Learning tels que le LSTM et le Bi-LSTM qui constitue la base de notre système. Enfin, nous avons cité quelques travaux récents dans le domaine de gestion de stock automatisé en particulier la détection en temps réelle et la prévision de la demande. Le chapitre suivant sera consacré à l'analyse et à la conception de notre système, dédié à la détection et à la prédiction des produits pharmaceutiques.

Chapitre 2

Analyse et conception du système

2.1 Introduction

Dans ce chapitre, nous présentons l'analyse des besoins nécessaire à la conception du **PharmaVision 4.0**, un système intelligent permettant la détection et la prévision en temps réel des médicaments. Nous commençons par l'élaboration d'un questionnaire destiné aux pharmaciens et aux vendeurs en pharmacie afin d'analyser l'existant et d'identifier les besoins fonctionnels liés à la gestion des stocks pharmaceutiques. Les résultats obtenus servent de base à l'analyse fonctionnelle du système proposé.

Ensuite, nous présentons l'architecture générale de la solution ainsi que ses couches matérielles et logicielles. Afin de décrire le fonctionnement du système et les interactions entre ses différents acteurs, nous élaborons des diagrammes de cas d'utilisation et des diagrammes de classes. Enfin, nous terminons ce chapitre par une conclusion.

2.2 Analyse des besoins

Pour la conception du système PharmaVision 4.0 qui répond aux besoins réels du terrain, nous avons élaboré un questionnaire destiné aux pharmaciens d'officine. Ce questionnaire, composé de dix questions, a été diffusé en ligne via Google Forms auprès de pharmaciens de la région de Tlemcen. Les questions ont été organisées selon trois axes : le premier porte sur les pratiques actuelles et les difficultés rencontrées (ruptures de stock, méthodes de gestion, erreurs d'inventaire), le deuxième évalue le besoin en outils de prévision de la demande, et le troisième mesure l'intérêt des professionnels pour les fonctionnalités envisagées dans notre système (détection automatique par caméra, alertes de rupture, tableau de bord en temps réel, recours à l'intelligence artificielle). Les résultats obtenus pour chaque question sont présentés ci-dessous. Le questionnaire utilisé dans cette étude est présenté en **Annexe A**.

Question 1 : Rencontrez-vous des ruptures de stock ?

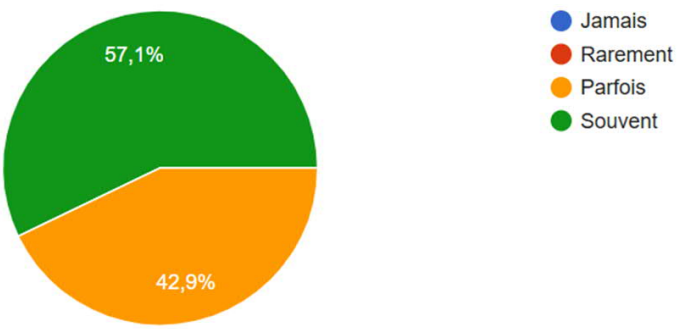


FIGURE 2.1 – Réponse de la question 1

Les résultats montrent que 57,1% des pharmaciens déclarent rencontrer souvent des ruptures de stock, tandis que 42,9% en rencontrent parfois. Autrement dit, tous les pharmaciens qui ont répondu sont touchés par ce problème, à des degrés différents. Aucun n'a répondu par « jamais » ou « rarement », cela montre bien à quel point les ruptures de stock sont fréquentes et préoccupantes dans la gestion quotidienne des stocks.

Question 2 : Comment gérez-vous actuellement votre stock ?

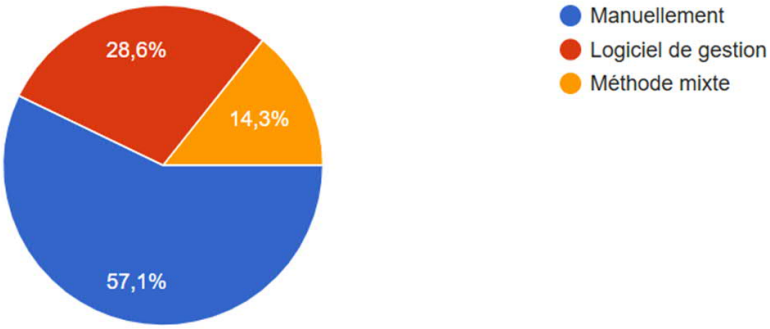


FIGURE 2.2 – Réponse de la question 2

Les résultats montrent que 28,6% des pharmaciens s'appuient sur les logiciels de gestion classique et 57,1% sur l'inventaire manuel et le reste pour la gestion de stock, ce qui montre que le suivi repose encore sur des saisies manuelles sujettes aux erreurs et aux oublis.

Question 3 : Les erreurs d'inventaire sont-elles fréquentes ?

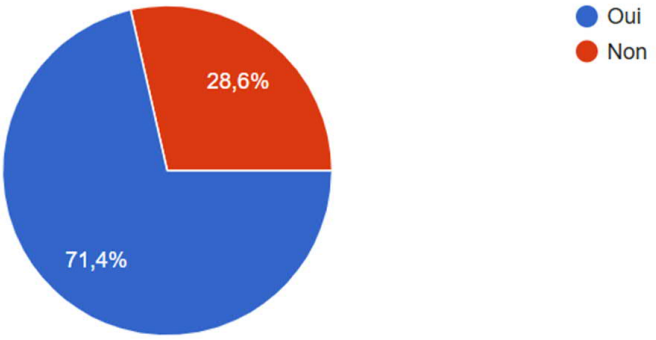


FIGURE 2.3 – Réponse de la question 3

Les résultats montrent que 71,4% des pharmaciens reconnaissent des erreurs d'inventaire fréquentes, traduisant l'écart entre le stock théorique enregistré et le stock physique réel.

Question 4 : La prévision de la demande est-elle importante pour votre activité ?

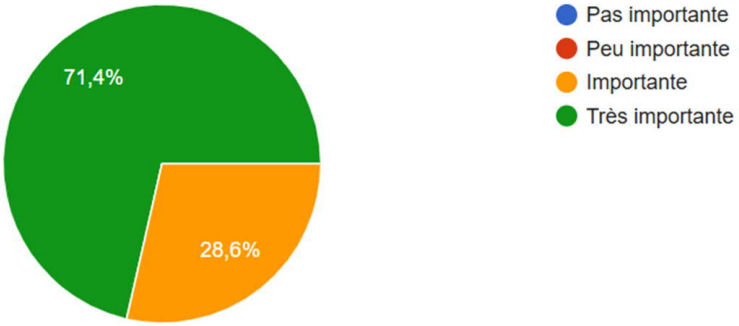


FIGURE 2.4 – Réponse de la question 4

Les résultats montrent que la prévision de la demande est importante (28,6%), voire très importante (71,4%), pour les pharmaciens afin d'anticiper les commandes et éviter les ruptures.

Question 5 : Utilisez-vous actuellement des outils de prévision ?

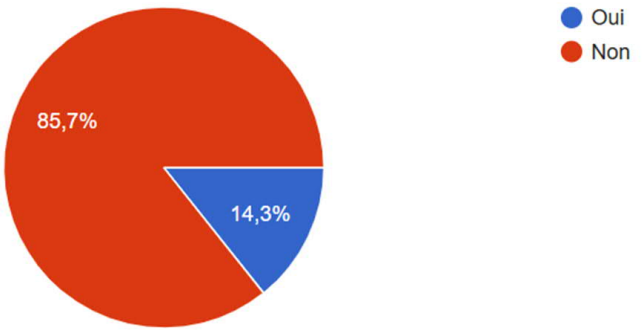


FIGURE 2.5 – Réponse de la question 5

Les résultats montrent que 85,7% des pharmaciens n'utilisent aucun outil de prévision, contre seulement 14,3% qui en disposent. Cela montre que les méthodes de prévision sont encore très peu employées dans la gestion des stocks.

Question 6 : Une détection automatique des médicaments par caméra pourrait-elle être utile ?

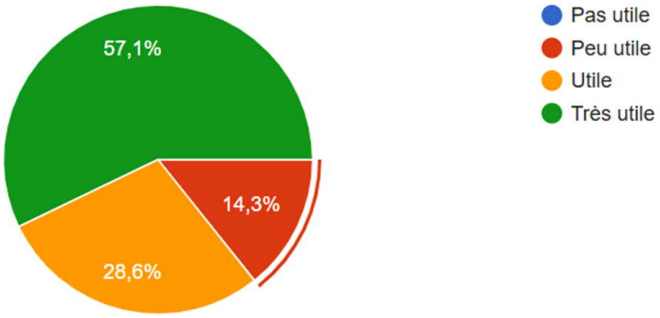


FIGURE 2.6 – Réponse de la question 6

Les résultats montrent que 57,1% des pharmaciens jugent la détection automatique par caméra très utile et 26,6% utile. Cela fait 83,7% d'avis favorables, une belle preuve que le terrain est prêt à adopter cette technologie.

Question 7 : Un système d'alerte automatique des ruptures de stock serait-il intéressant ?

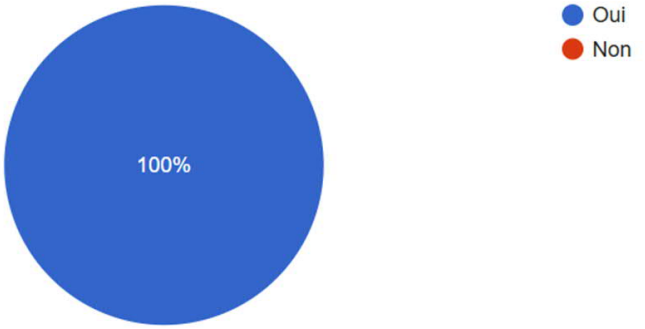


FIGURE 2.7 – Réponse de la question 7

Les résultats montrent que tous les pharmaciens interrogés trouvent intéressant un système d'alerte automatique en cas de rupture.

Question 8 : Un tableau de bord en temps réel faciliterait-il votre travail ?

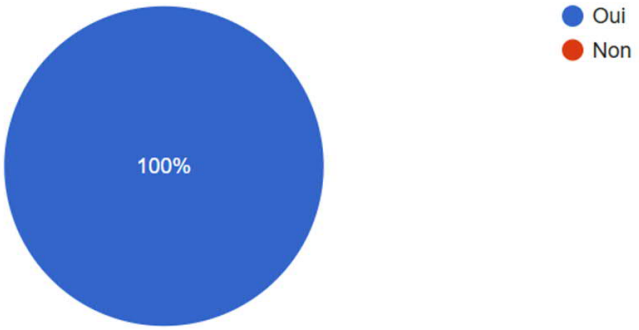


FIGURE 2.8 – Réponse de la question 8

Même résultat pour le tableau de bord en temps réel, 100% des répondants pensent qu'il leur faciliterait le travail, confirmant qu'il y a une demande forte pour des outils de suivi instantané et centralisé.

Question 9 : Quels indicateurs aimeriez-vous suivre ?

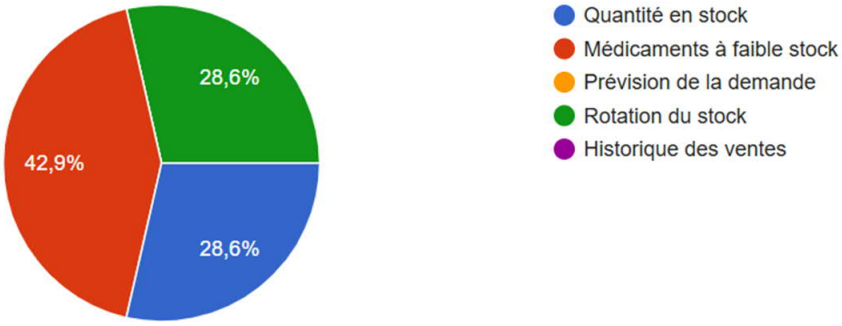


FIGURE 2.9 – Réponse de la question 9

Ce que les pharmaciens veulent surveiller en priorité : 42,9% s'intéressent d'abord aux médicaments en stock faible, puis 28,6% à la quantité en stock et 28,6% à la rotation. On remarque que l'attention se porte surtout sur les niveaux critiques.

Question 10 : Seriez-vous favorable à l'utilisation d'une solution basée sur l'IA pour la gestion des stocks ?

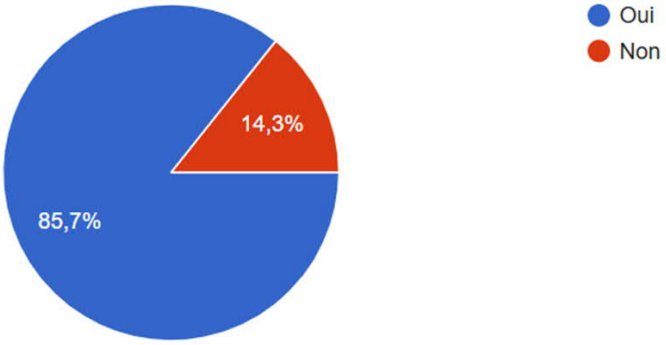


FIGURE 2.10 – Réponse de la question 10

Une grande majorité des pharmaciens interrogés (85,7%) se montre ouverte à l'idée d'adopter une solution basée sur l'IA pour gérer leurs stocks, ce qui témoigne d'une réelle volonté d'évolution vers des outils plus intelligents.

D'après l'analyse des besoins, nous remarquons que la gestion des stocks pharmaceutiques repose encore sur des méthodes manuelles ou des logiciels classiques, qui exigent une saisie humaine à chaque mouvement et ne fournissent ni alerte proactive ni capacité de prévision en temps réel.

Le système PharmaVision 4.0 que nous proposons se distingue des solutions existantes sur trois plans. Premièrement, là où les logiciels de gestion classiques dépendent des saisies du pharmacien, notre système observe directement le stock physique grâce à des caméras et à la vision par ordinateur : l'inventaire reflète ce qui se trouve réellement sur les étagères, et non ce qui a été déclaré.

Deuxièmement, la détection des ruptures n'est plus constatée à posteriori mais signalée automatiquement et en temps réel par un mécanisme d'alertes. Troisièmement, notre solution intègre une dimension prédictive, absente des outils actuellement utilisés par les pharmaciens interrogés, afin d'anticiper la demande plutôt que de la subir.

Notre travail se positionne ainsi à l'intersection de trois domaines : la vision par ordinateur (détection et comptage par YOLOv8), les systèmes embarqués (traitement local sur Raspberry Pi) et l'aide à la décision (tableau de bord de supervision et prévision de la demande). Chacune des fonctionnalités retenues répond directement à un besoin identifié dans l'enquête, ce qui garantit l'adéquation entre la solution conçue et les attentes du terrain. La section suivante présente l'architecture générale de ce système.

2.3 Architecture générale du système

Le système développé, nommé PharmaVision 4.0, est une maquette intelligente de supervision des stocks de médicaments. L'architecture du système repose sur trois composantes principales :

— **Deux Raspberry Pi 3 Model B** : Chaque rack est surveillé par un Raspberry Pi 3 Model B équipé d'une caméra Raspberry Pi Camera V2. Les caméras capturent périodiquement des images des médicaments présents sur les étagères. Les Raspberry Pi exécutent localement un modèle de vision par ordinateur basé sur YOLOv8 afin de détecter et de compter les boîtes de médicaments.

— **Un serveur central** : Le serveur central est exécuté sur l'ordinateur du pharmacien. Il collecte les informations provenant des deux Raspberry Pi, fusionne les inventaires des deux racks, détecte les variations de stock et enregistre l'ensemble des données dans une base de données SQLite. Il fournit également une API REST permettant l'accès aux données du système.

— **Un tableau de bord web** : Le tableau de bord constitue l'interface de supervision du système. Il permet au pharmacien de consulter en temps réel l'état du stock des médicaments, de visualiser les alertes de stock faible ou de rupture, d'accéder à l'historique des mouvements et d'exporter les données si nécessaire.

2.3.1 Description de la couche matérielle

La couche matérielle regroupe l'ensemble des composants physiques utilisés pour la réalisation de la maquette du système PharmaVision 4.0. Elle est constituée d'une plaque en bois représentant le sol du système, de quatre racks de stockage, de boîtes de médicaments, de deux Raspberry Pi 3 Model B équipés de caméras Raspberry Pi Camera V2, de boîtiers de fixation imprimés en 3D et d'un PC pour la visualisation du tableau de bord et des résultats de surveillance en temps réel. Le prototype physique réalisé pour valider notre système est illustré par la figure 2.11.



FIGURE 2.11 – Prototype du projet de fin d'étude.

Plaque en bois

La maquette repose sur une plaque en bois représentant le sol du système. Cette plaque constitue la base de l'installation et supporte deux racks A et B disposés parallèlement. Chaque rack comporte trois niveaux de stockage permettant de déposer et d'organiser les boîtes de médicaments.

Boîtes de médicaments

Nous avons choisi 18 types de médicaments et chaque type contient 4 boîtes pour tester le système PharmaVision 4.0.

Les références sélectionnées sont : Approvel, Clamoxyl 1 g, Domperone, Etofan, Flecar, Levothyrox, Molsidomine, Peptidac, Phalugel, Rhumafed, Sebutol, Smedyl, Tareg 160, Trimebutine B, Xydol 400, Zopra, Zostine et Vitamine C UPSA 1 g comprimés effervescents (boîte de 20). Pour chaque type de médicament, trois boîtes ont été intégrées à la maquette, ce qui représente un stock initial total de 54 boîtes. Cette diversité de références permet de reproduire un environnement de stockage réaliste et d'évaluer les performances du système de suivi, de détection et de prévision des stocks.

Raspberry Pi

Nous aurions pu nous limiter à un seul Raspberry Pi et à un seul rack : cela aurait suffi à démontrer la faisabilité technique. Mais notre objectif était de concevoir un système capable de s'adapter à un véritable entrepôt comportant de nombreuses zones de stockage, et une seule caméra ne suffit jamais à tout couvrir. C'est pourquoi nous avons choisi d'installer deux Raspberry Pi, chacun associé à sa propre zone de stockage. L'originalité de cette configuration ne réside pas dans la simple multiplication des caméras, mais dans le fait qu'elles travaillent ensemble pour couvrir l'espace de stockage : le système fusionne ensuite leurs résultats afin de produire un inventaire unique, reflétant l'état complet du stock. C'est cette coopération qui rend l'architecture réellement exploitable dans un cas réel.

Pour la partie traitement et communication, nous avons choisi deux cartes Raspberry Pi 3 Model B. Ce modèle a été lancé en février 2016. Selon la fiche technique officielle du fabricant, ses caractéristiques principales sont les suivantes :



FIGURE 2.12 – Raspberry Pi 3 Model B

— Un processeur Broadcom BCM2837 quad-cœur 64 bits cadencé à 1,2 GHz.

— 1 Go de RAM (mémoire vive suffisante pour faire tourner un système d'exploitation léger et un programme de détection YOLO).

— Une connectivité sans fil au réseau Wi-Fi sans câble Ethernet, ce qui a simplifié l'installation.

— Un port Ethernet pour la connexion filaire si nécessaire.

— 4 ports USB 2.0 (pour brancher des périphériques comme souris et clavier).

— Un port HDMI plein format pour la connexion à un écran.

— Une prise audio/vidéo.

— Un connecteur GPIO étendu à 40 broches.

— Un port CSI pour connecter directement la caméra Raspberry, indispensable pour ce projet.

— Un port DSI pour un écran tactile, et un port microSD pour le système d'exploitation et le stockage des données.

— L'alimentation se fait via un connecteur Micro USB pouvant supporter jusqu'à 2,5 A [18].

2.3.2 Caméra

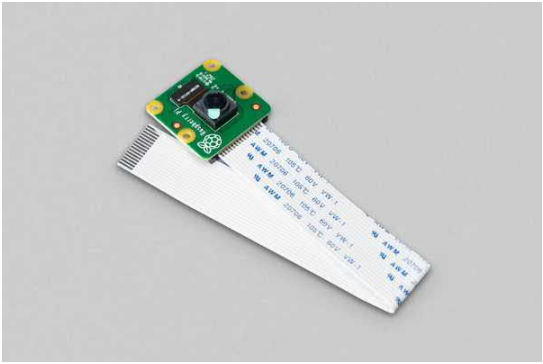


FIGURE 2.13 – Caméra Raspberry Pi module v2

Chacun des deux Raspberry Pi est équipé d'une caméra officielle Raspberry Pi Camera Module 2 qui se connecte directement au port CSI (*Camera Serial Interface*) de la carte via un câble nappe. La Camera Module 2 embarque un capteur Sony IMX219 de 8 mégapixels. Ses caractéristiques principales sont :

— Résolution maximale : 3280 × 2464 pixels.

— Taille de pixel : 1,12 µm.

— Angle de vue : environ 62 degrés (avec l'objectif standard) [19].

Les deux Raspberry Pi sont installés au-dessus des racks ; chaque caméra, placée à environ 50 cm de hauteur, couvre ainsi l'intégralité du rack qui lui est associé. Cette hauteur, combinée à l'angle de vue de 62 degrés du capteur, garantit que l'ensemble des trois niveaux de stockage entre dans le champ de vision.

Boîtier du Raspberry Pi

Pour fixer chaque Raspberry Pi au-dessus des racks, nous avons utilisé un boîtier imprimé en 3D, adapté d'un modèle conçu initialement pour le Raspberry Pi 4. Ce boîtier permet de maintenir la caméra et de l'orienter selon l'angle souhaité.

2.3.3 Description de la couche logicielle

Le système est divisé en quatre briques logicielles indépendantes :

— **Programme de détection tournant sur les deux Raspberry Pi (main.py)** : il capture les images, effectue l'inférence, puis propose un serveur Flask local donnant accès aux routes /video_feed et /inventory.

— **Serveur central (server.py)** : collecte périodiquement les résultats des Raspberry Pi et les fusionne pour former un inventaire complet, note les changements de stock, sauvegarde les données dans SQLite et offre une API REST au tableau de bord. Il expose également des routes dédiées au tableau des ventes mensuelles et au déclenchement de l'entraînement du modèle de prédiction.

— **Module de prédiction** : entraîné à partir des données de ventes des trois derniers mois exportés par le serveur central, il génère un fichier Excel contenant les quantités prédites pour le mois suivant, lequel est lu par le tableau de bord pour afficher les résultats.

— **Tableau de bord (dashboard.html)** : interface utilisateur en HTML/CSS/JS qui interroge le serveur central et affiche les flux vidéo, l'inventaire, les alertes, les ventes et les prédictions de demande. Le flux de données du système suit un parcours en quatre étapes :

1. Les Raspberry Pi exposent leurs inventaires en JSON.
2. Le serveur central les interroge, calcule les variations et met à jour la base.
3. En fin de mois, le serveur exporte les données de ventes et déclenche l'entraînement du modèle BiLSTM, qui produit un fichier de prédictions pour le mois suivant.
4. Le tableau de bord interroge le serveur central pour afficher les données actuelles et consulte le fichier de prédictions pour la planification des stocks.

Le programme s'appuie sur plusieurs bibliothèques Python, dont les rôles sont résumés dans le tableau 2.1.

TABLE 2.1 – Bibliothèques logicielles utilisées

Catégorie	**Technologie/ Bibliothèque**	**Rôle**
Langage	Python 3	Langage utilisé pour les programmes Raspberry Pi et serveur central. Facile, riche en bibliothèques et adapté à la vision par ordinateur.
Framework web	Flask	Serveur web. Sur les Raspberry Pi, il diffuse le flux vidéo et l'inventaire ; sur le serveur central, il fournit l'API REST pour le tableau de bord.
Détection	YOLOv8/NCNN	YOLOv8 entraîné sur la base de données pour la reconnaissance des médicaments, puis converti au format NCNN pour une inférence rapide et adaptée au Raspberry Pi.
Base de données	SQLite	Stockage persistant de l'historique, des alertes et du dernier inventaire dans un fichier (pharma.db).
Communication	HTTP + JSON	Échange de données entre les Raspberry Pi, le serveur central et le tableau de bord via une API REST simple.
Traitement d'image	OpenCV (cv2)	Redimensionnement, conversion de couleurs, dessin des rectangles, NMS (*Non-Maximum Suppression*).
Capture vidéo	Picamera2	Interface officielle pour la caméra du Raspberry Pi, permettant le réglage d'exposition et la capture en continu.
Calcul numérique	NumPy	Extraction des informations depuis les tenseurs produits par le modèle : coordonnées des boîtes, scores de confiance, classes des médicaments (post-traitement des détections).
Parallélisme	Threading	Exécution simultanée de la boucle de détection et du serveur web (sur chaque Raspberry Pi), et collecte des données en arrière-plan (sur le serveur central).
Apprentissage profond	PyTorch	Framework utilisé pour construire et entraîner le modèle Bi-LSTM de prévision de la demande.
Calcul numérique (prévision)	NumPy	Manipulation des tableaux de données et calcul des métriques d'évaluation du modèle Bi-LSTM.
Lecture des fichiers Excel	xlrd/openpyxl	Lecture des fichiers XLS contenant les registres mensuels de ventes de la pharmacie.
Manipulation des données	Pandas	Lecture des 41 fichiers XLS, construction du tableau pivot mensuel et préparation des séquences d'entraînement.
Interface tableau de bord	HTML/CSS/ JavaScript	Développement du tableau de bord PharmaVision 4.0 sans framework externe, avec affichage en temps réel des inventaires, alertes et historiques.
Annotation et préparation des données	Roboflow	Plateforme utilisée pour annoter manuellement les 772 images, appliquer des transformations d'augmentation (rotation, luminosité, etc.) et exporter le dataset au format YOLO.
Environnement d'entraînement	Google Colab (GPU T4)	Plateforme utilisée pour entraîner le modèle Bi-LSTM avec un GPU T4 de 16 Go de VRAM.
Lecture/écriture Excel	SheetJS (xlsx.js)	Lecture du fichier de prédictions BiLSTM et génération des exports Excel directement dans le navigateur du tableau de bord.

Dans cette partie, nous avons présenté l'architecture générale du système PharmaVision 4.0. Afin de mieux illustrer son fonctionnement et de comprendre les interactions entre ses composants matériels et logiciels, nous avons opté pour une modélisation UML. Cette approche permet de structurer la description du système et de représenter de manière formelle ses différents comportements sous forme de diagramme de cas d'utilisation et de diagrammes de séquence.

2.4 Modélisation UML

2.4.1 Diagramme de cas d'utilisation

Ce diagramme illustre le cas d'utilisation du système PharmaVision 4.0 ; il comporte deux volets : un volet automatisé (capture, détection, comptage, mise à jour, alerte) porté par les acteurs techniques, et un volet de supervision porté par le pharmacien, qui exploite en temps réel les données produites par l'intelligence artificielle. Cette architecture réduit les erreurs de saisie, supprime les inventaires manuels et garantit une visibilité continue sur l'état du stock pharmaceutique.

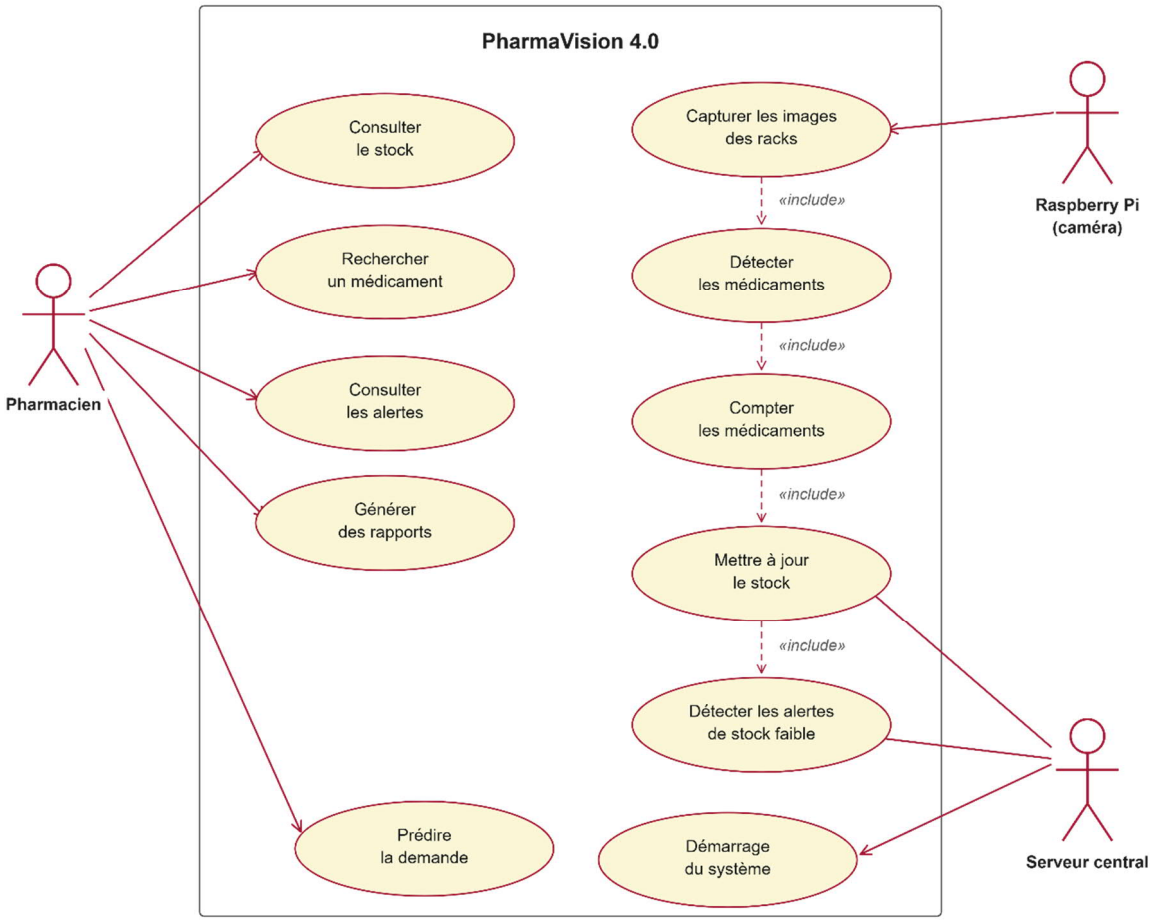


FIGURE 2.14 – Diagramme de cas d'utilisation du système PharmaVision 4.0

Cas d'utilisation « Démarrage du système »

— Le serveur démarre et charge depuis la base de données le dernier inventaire connu, l'historique des mouvements et les alertes actives, et les place en mémoire.

— Le système reprend exactement dans l'état où il se trouvait avant l'arrêt.

— La boucle de détection est lancée.

Cas d'utilisation « Consulter le stock »

— Le pharmacien accède à la rubrique « Stock ».

— Le système interroge la table dernière_inventaire de la base de données et récupère la liste des médicaments avec leurs quantités actuelles.

— Le système affiche la liste sous forme de tableau (nom, quantité, date de dernière mise à jour, statut).

Cas d'utilisation « Rechercher un médicament »

— Le pharmacien saisit le nom du médicament recherché.

— Le système interroge la base de données et identifie les références correspondantes.

— Le système affiche la liste des résultats : pour chaque médicament, quantité disponible, statut et historique récent des mouvements.

— Le pharmacien sélectionne un médicament pour afficher sa fiche détaillée.

Cas d'utilisation « Consulter les alertes »

— Le système notifie le pharmacien lorsqu'une alerte est générée.

— Le pharmacien ouvre le centre des alertes.

— Le système interroge la table alertes et affiche la liste des alertes actives.

— Le pharmacien consulte le détail d'une alerte (médicament concerné, quantité restante, type d'alerte et date de détection).

Cas d'utilisation « Générer des rapports »

— Le pharmacien choisit le type de rapport (inventaire, historique des mouvements ou alertes).

— Le système interroge la base de données et génère le rapport.

— Le système affiche le rapport et propose son export.

— Le pharmacien consulte ou exporte le rapport.

Cas d'utilisation « Capturer les images des racks »

— Chaque Raspberry Pi capture en continu les images du rack qui lui est associé via sa caméra.

— Le module YOLO embarqué sur le Raspberry Pi effectue la détection et le comptage localement sur l'image capturée.

— Les résultats de comptage sont exposés en JSON via /inventory du Raspberry Pi.

— Le cas d'utilisation « Détecter les médicaments » est déclenché en continu.

Cas d'utilisation « Détecter les médicaments »

— Le module YOLO embarqué reçoit l'image capturée et applique les prétraitements nécessaires (redimensionnement à 640×640, normalisation).

— Le modèle effectue l'inférence et produit, pour chaque boîte détectée, une boîte englobante, une classe (référence du médicament) et un score de confiance.

— Le module filtre les détections dont le score est inférieur au seuil de confiance défini (CONF = 0,35).

— Les détections validées sont transmises au comptage.

Cas d'utilisation « Compter les médicaments »

— Le module YOLO regroupe les détections validées par classe de médicament.

— Le module calcule le nombre de boîtes détectées par référence pour le rack associé.

— Les comptages sont exposés via /inventory et mis à disposition du serveur central.

Cas d'utilisation « Mettre à jour le stock »

— Le serveur central interroge toutes les 3 secondes les deux Raspberry Pi et récupère les comptages.

— Il fusionne les deux inventaires en mémoire et compare les nouvelles quantités avec inventaire_precedent.

— Pour chaque variation détectée, il met à jour inventaire_precedent en mémoire pour le cycle suivant.

— En parallèle, il sauvegarde les nouvelles quantités dans la table dernier_inventaire et journalise chaque écart dans la table historique, afin de garantir la reprise correcte en cas d'arrêt du système.

— La vérification des seuils d'alerte est déclenchée.

Cas d'utilisation « Détecter les alertes de stock faible »

— Le serveur compare chaque quantité à son seuil.

— Quantité nulle : alerte « rupture de stock ».

— Quantité $\leq$ seuil : alerte « alerte IA ».

— Sinon : stock disponible, aucune alerte.

— La table des alertes est effacée puis reconstruite avec les alertes actives du actuel.

— Les alertes sont affichées sur le tableau de bord, et le pharmacien est notifié.

Cas d'utilisation « prédire la demande »

— Le pharmacien demande une prédiction pour le mois suivant.

— Le système calcule les ventes de M, M-1 et M-2 depuis l'historique.

— Le système exporte ces données dans des fichiers Excel.

— Le système lance automatiquement le modèle BiLSTM en arrière-plan.

— Le modèle s'entraîne sur les 3 mois de données et génère une prédiction.

— Le système sauvegarde le résultat sous forme de fichier Excel (exemple : prediction_mois_année.xlsx).

— Le pharmacien télécharge le fichier de prédiction souhaité.

Diagrammes de séquence

Les diagrammes de séquence présentés ci-dessous permettent de représenter le déroulement de quelques scénarios du système PharmaVision 4.0.

Le diagramme de séquence ci-dessous (figure 2.15) représente le scénario de synchronisation de l'inventaire entre les Raspberry Pi, le serveur central, la base de données et le tableau de bord. Toutes les trois secondes, le serveur central interroge les deux Raspberry Pi afin de récupérer les comptages des médicaments détectés dans les racks sous forme de données JSON. Le serveur fusionne ensuite les inventaires reçus pour construire une vue globale du stock. Il compare cet inventaire avec le précédent afin d'identifier les variations, puis met à jour les informations dans la base de données et enregistre l'historique des écarts. Après vérification des seuils de stock, le système reconstruit la table des alertes et notifie les alertes actives au tableau de bord. Ce processus permet d'assurer un suivi continu et en temps réel de l'état du stock.

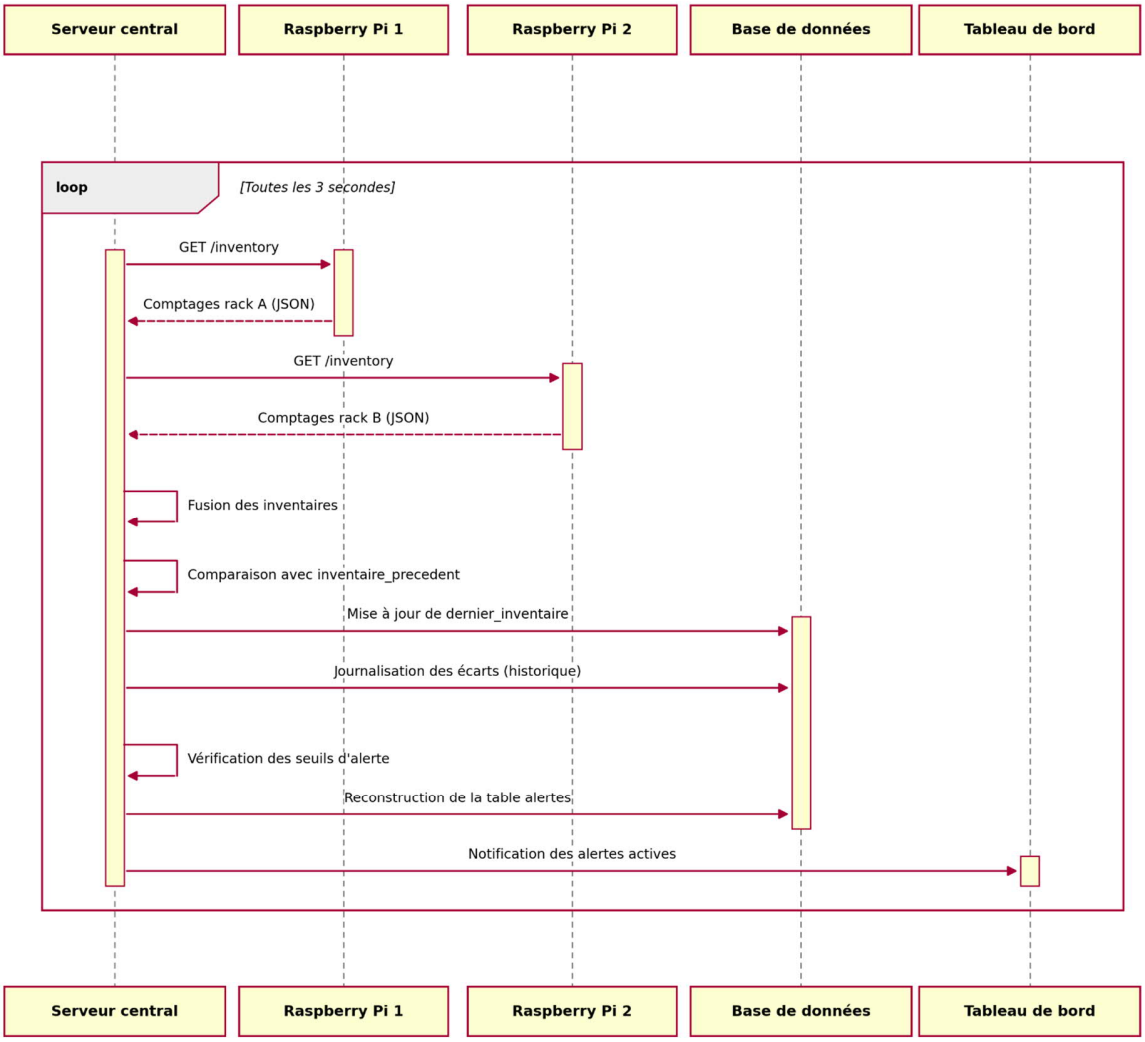


FIGURE 2.15 – Diagramme de séquence de mise à jour de l'inventaire et de gestion des alertes de PharmaVision 4.0

Le diagramme ci-dessous (Figure 2.16) illustre le processus de consultation des alertes de stock dans le système PharmaVision 4.0. Lorsqu'une nouvelle alerte est générée, une notification est affichée sur le tableau de bord. Le pharmacien accède alors au centre des alertes, ce qui déclenche une requête vers le serveur central. Celui-ci interroge la base de données afin de récupérer les alertes actives et transmet les résultats au tableau de bord. Le pharmacien peut ensuite consulter la liste des alertes et afficher les détails d'une alerte sélectionnée.

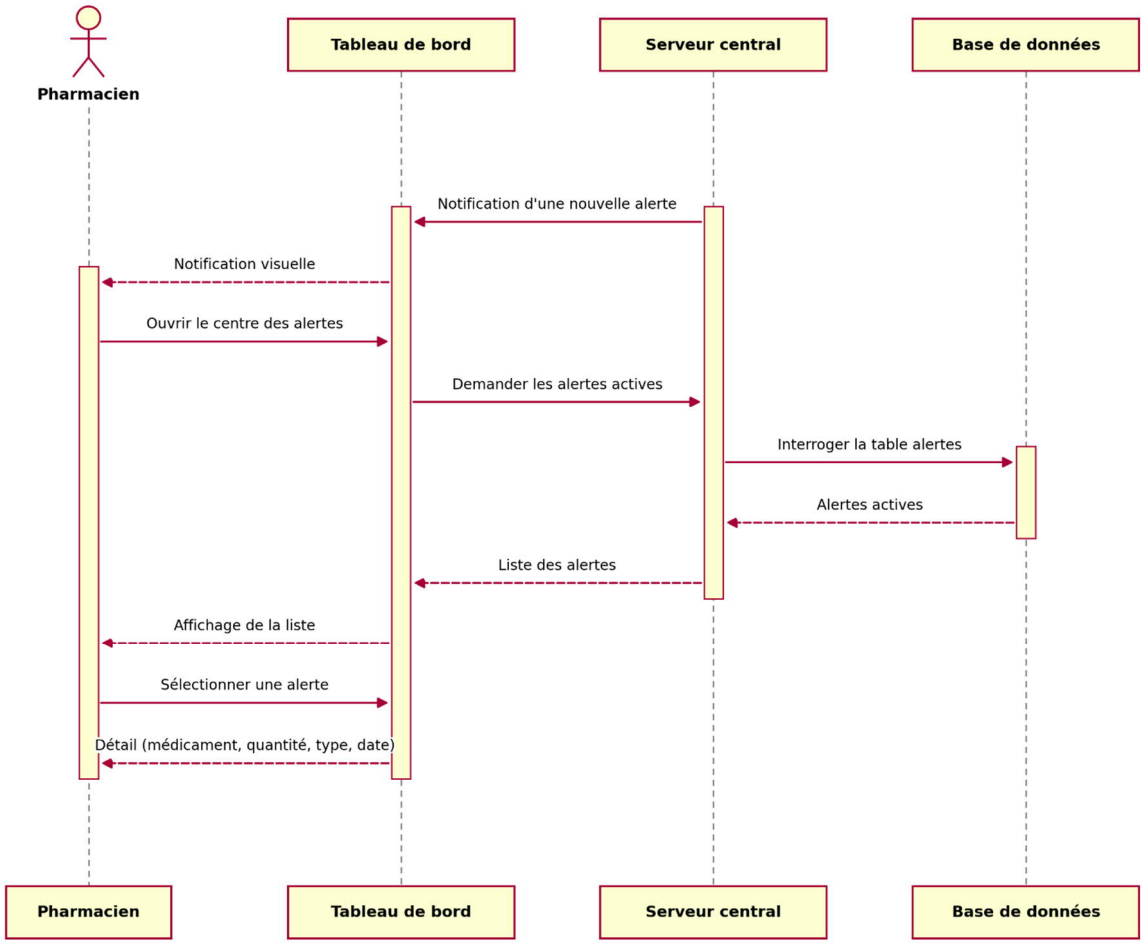


FIGURE 2.16 – Diagramme de séquence de la consultation des alertes via le tableau de bord

Le diagramme ci-dessous (Figure 2.17) illustre le processus de consultation du stock dans le système PharmaVision 4.0. Le pharmacien accède à la rubrique « Stock » depuis le tableau de bord. Celui-ci transmet alors la demande au serveur central en sollicitant l'état du stock. Le serveur central interroge ensuite la table dernier_inventaire de la base de données, qui lui retourne la liste des médicaments accompagnée de leurs quantités actuelles. Ces informations remontent ensuite au tableau de bord, qui les affiche au pharmacien sous forme de tableau présentant, pour chaque médicament, son nom, sa quantité, sa date de mise à jour et son statut.

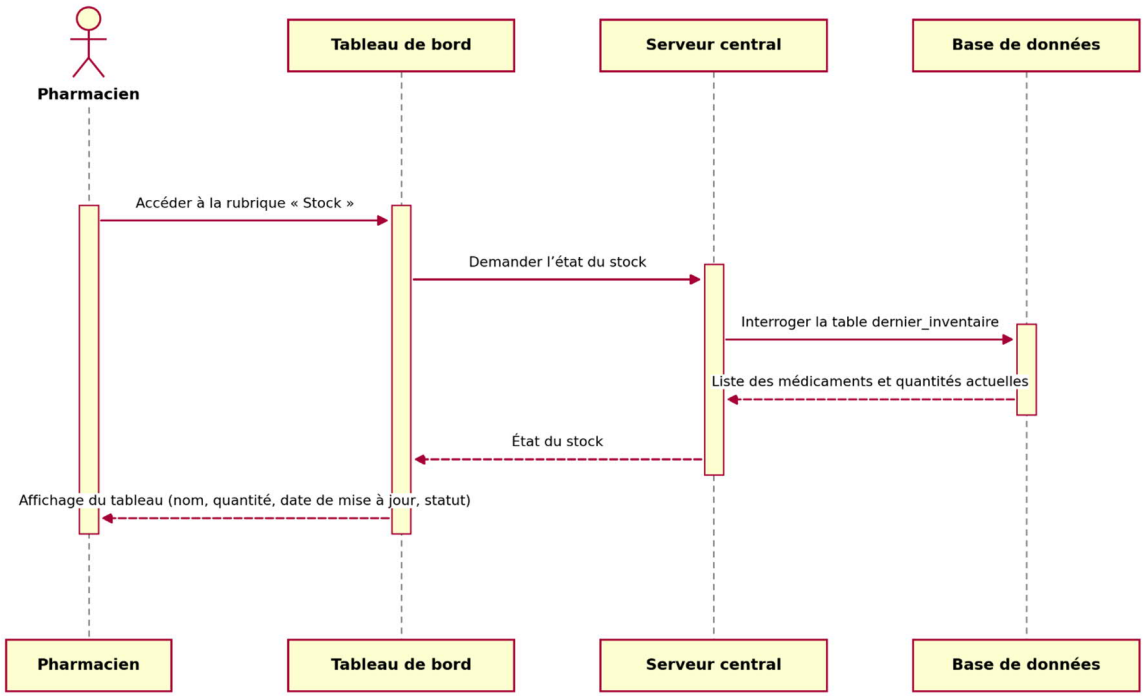


FIGURE 2.17 – Diagramme de séquence « Consulter le stock »

Le diagramme de séquence ci-dessous (Figure 2.18) décrit la boucle de détection qui s'exécute en continu sur chaque Raspberry Pi de PharmaVision 4.0. Il fait intervenir trois participants : la caméra (représentée comme acteur déclencheur), le Raspberry Pi (le programme principal main.py) et le module YOLO (le modèle YOLOv8 converti au format NCNN, exécuté localement sur la carte).

La caméra capture une image du rack et la transmet au programme du Raspberry Pi via l'interface Picamera2. Le Raspberry Pi transmet ensuite l'image capturée au module YOLO, qui effectue deux traitements internes représentés par des auto-messages : la détection des médicaments (identification des boîtes présentes sur l'image) puis le comptage par référence (dénombrement des boîtes pour chaque médicament reconnu).

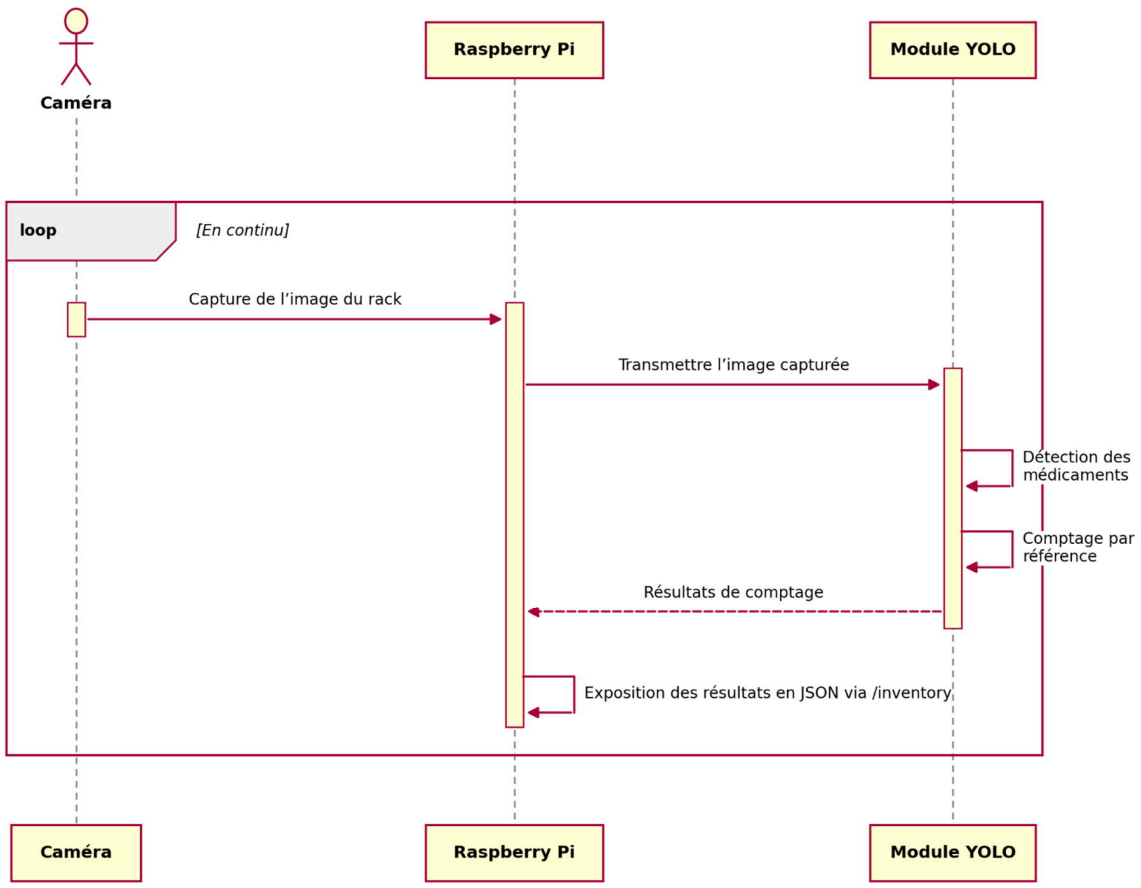


FIGURE 2.18 – Diagramme de séquence du processus de détection et de comptage

Le diagramme de séquence ci-dessous (Figure 2.19) illustre le déroulement du processus de prédiction de la demande basé sur le modèle Bi-LSTM. Lorsque le pharmacien souhaite anticiper les demandes du mois prochain, il déclenche la procédure depuis le tableau de bord. Le serveur central commence par interroger la base de données historique afin de récupérer les données de ventes des trois derniers mois. Ces données sont ensuite exportées sous forme de fichiers Excel, qui servent d'entrée au modèle Bi-LSTM pour lancer l'entraînement en arrière-plan. Une fois l'entraînement terminé, le modèle génère les prédictions et le fichier résultant est sauvegardé sous le nom prediction_Mois_année.xlsx. Le serveur notifie alors le pharmacien que les prédictions sont prêtes, lui permettant de télécharger directement le fichier et de consulter les résultats depuis l'interface.

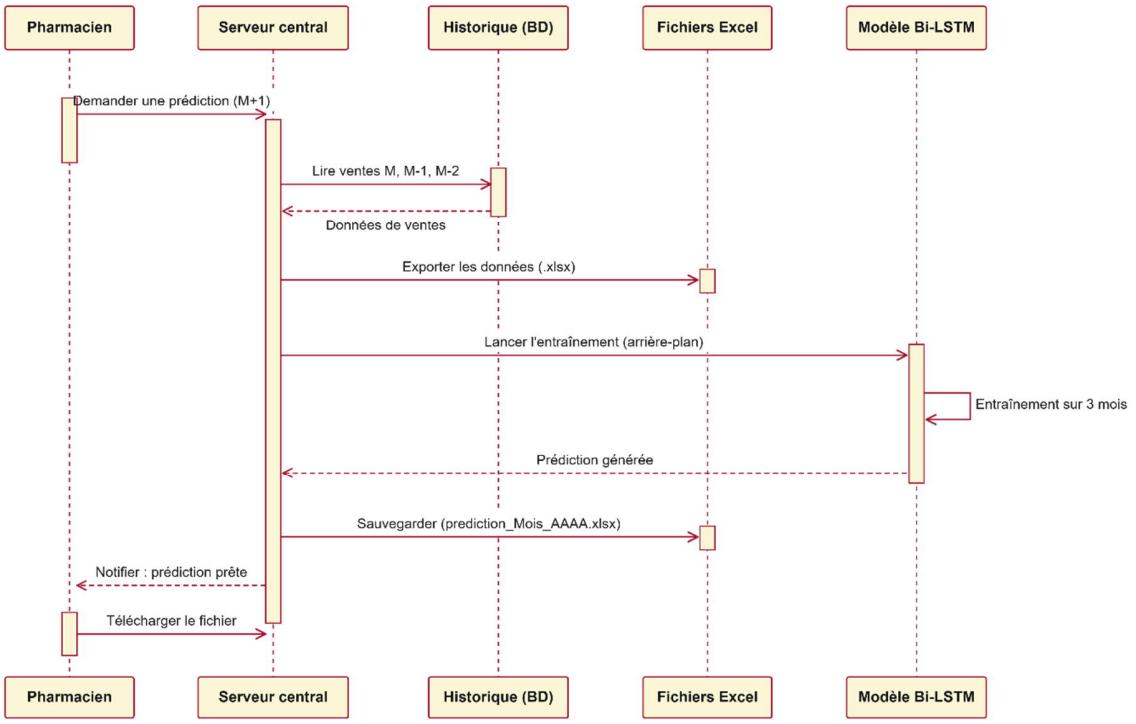


FIGURE 2.19 – Diagramme de séquence du processus de prédiction de la demande (modèle Bi-LSTM)

2.5 Scénario de fonctionnement du système PharmaVision 4.0

2.5.1 Capture d'image

Chacun des deux Raspberry Pi capture en temps réel des images via son module Picamera2 à une résolution de 1280×720 pixels au format BGR. La caméra est configurée avec un recadrage maximal afin d'exploiter toute la surface du capteur et d'obtenir la meilleure netteté possible sur les détails fins des boîtes de médicaments. Une image est capturée toutes les 10 ms pour garantir une surveillance quasi-continue du stock.

2.5.2 Détection des produits

Avant d'être soumise au modèle d'intelligence artificielle, chaque image doit subir une étape de prétraitement. L'image est redimensionnée sans la déformer, puis placée au centre d'un carré gris de 640×640 pixels. Cette technique est appelée *letterboxing* : plutôt que d'étirer l'image pour qu'elle remplisse le carré (ce qui déformerait les boîtes de médicaments), elle la redimensionne proportionnellement et comble les bords vides par une couleur neutre.

Elle est ensuite normalisée, c'est-à-dire que les valeurs des pixels, initialement comprises entre 0 et 255, sont ramenées dans l'intervalle $[0, 1]$ par simple division par 255. Cette étape est indispensable car les réseaux de neurones apprennent plus vite et plus

facilement quand les valeurs d'entrée sont petites et homogènes.

L'image préparée est ensuite soumise au modèle YOLOv8 (*You Only Look Once*, version 8), un réseau de neurones convolutif spécialisé dans la détection d'objets en temps réel. Pour qu'il tourne sur le Raspberry Pi (qui ne dispose pas de carte graphique dédiée), il a été converti au format NCNN, un moteur d'inférence léger et optimisé pour les processeurs ARM comme celui du Raspberry Pi. Le modèle analyse l'image en une seule passe, et pour chaque objet repéré, il donne sa position (la boîte englobante) et un score de confiance indiquant à quel point il est sûr d'avoir identifié le bon médicament. Il traite chacune des 105 classes qu'il connaît. Autrement dit, pour chaque objet détecté, il se pose 105 questions en parallèle : est-ce un Clamoxyl ? un Dompéridone ? un Levothyrox ?... Et il attribue une probabilité à chaque réponse. Le médicament avec le score le plus élevé est retenu comme identification finale, à condition que ce score dépasse le seuil de 0,35, sinon la détection est rejetée pour éviter toute erreur. Enfin, l'algorithme NMS (*Non-Maximum Suppression*) élimine les boîtes en double lorsque plusieurs détections se chevauchent sur un même médicament.

2.5.3 Mise à jour du stock

Les détections validées sont stockées dans une variable partagée current_detections. Le serveur Flask, qui tourne en parallèle sur le même Raspberry Pi, peut y accéder en temps réel. Si un même médicament apparaît plusieurs fois devant la caméra, ses occurrences sont simplement additionnées. En même temps, l'image annotée est encodée en JPEG et stockée dans current_frame pour être diffusée en flux vidéo.

Toutes les 3 secondes, le serveur central interroge les deux Raspberry Pi pour récupérer leur inventaire respectif (Raspi 1 pour les racks A et Raspi 2 pour les racks B), puis additionne les quantités pour produire un inventaire global. Si l'un des Raspberry Pi ne répond pas, le système affiche une alerte pour informer l'utilisateur.

2.5.4 Stockage dans la base de données

À chaque cycle de collecte, le serveur central compare l'inventaire qu'il vient de recevoir avec celui du cycle précédent pour l'ensemble des 105 médicaments détectables. Dès qu'il détecte un changement (ajout ou retrait d'une boîte), il l'enregistre définitivement dans la table historique de la base SQLite pharma.db. Pour chaque variation, il note la date et l'heure exactes, la quantité avant, la quantité après, et la différence (positive si réapprovisionnement, négative si retrait).

La table dernière_inventaire est mise à jour à chaque cycle pour refléter l'état courant du stock. Elle sert, au redémarrage du serveur, à retrouver le dernier état connu, évitant ainsi que tous les médicaments détectés soient interprétés comme des réapprovisionnements, ce qui fausserait l'historique. Parallèlement, la table ventes cumule mois

par mois les sorties de stock, c'est-à-dire toutes les variations négatives enregistrées dans l'historique. Ces données constituent la base d'apprentissage du modèle Bi-LSTM. Le journal des entraînements, quant à lui, conserve une trace de chaque lancement du processus de prédiction : date du lancement, mois des données utilisées, mois cible prédit et statut de l'opération. Enfin, la table alertes est recalculée à chaque cycle. Elle ne contient que les médicaments en situation critique (en rupture ou en stock faible), ce qui permet de prévenir le pharmacien en temps réel dès qu'un problème survient.

2.5.5 Prédiction de la demande

En fin de mois, le pharmacien peut déclencher depuis le tableau de bord le processus de prédiction pour le mois suivant. Le serveur central récupère alors les données de ventes des trois derniers mois depuis la base SQLite et les exporte sous forme de fichiers Excel. Ces fichiers alimentent le modèle Bi-LSTM, qui s'entraîne en arrière-plan sur ces séquences temporelles et génère en sortie un fichier prediction_Mois_année.xlsx contenant les quantités prédites pour chaque médicament. Une fois l'entraînement terminé, le pharmacien est notifié et peut consulter directement depuis l'onglet Prédiction IA les résultats sous forme de tableau et de graphiques, afin d'anticiper les besoins en stock et d'optimiser ses commandes.

2.5.6 Affichage sur le tableau de bord

Le tableau de bord dashboard.html est accessible depuis le navigateur du PC du pharmacien. Il interroge automatiquement le serveur central toutes les 3 secondes puis affiche en temps réel, dans l'accueil, quatre indicateurs clés : le nombre total de médicaments détectés, le nombre de références actives, le nombre de caméras connectées et le nombre d'alertes en cours. Le pharmacien dispose d'une vue complète sur les 105 références de médicaments avec leur statut (disponible, critique ou épuisé), d'un moteur de recherche et de filtres pour naviguer rapidement dans le stock, ainsi que de l'historique complet des variations avec code couleur (vert pour les entrées, rouge pour les sorties). Les flux vidéo en direct des deux caméras sont accessibles depuis l'onglet Caméras. En cas de stock critique, une bannière d'alerte rouge s'affiche automatiquement et un badge indique le nombre d'alertes actives. L'ensemble des données peut être exporté à tout moment en format CSV ou TXT pour archivage ou transmission. Depuis l'onglet Prédiction IA, le pharmacien peut déclencher en fin de mois le processus d'entraînement du modèle Bi-LSTM et suivre sa progression en temps réel grâce à une barre d'avancement. Une fois l'entraînement terminé, il consulte les prévisions de demande pour le mois suivant sous forme de tableau détaillé et de graphiques, avec pour chaque médicament la quantité prédite et son niveau de demande (forte, modérée ou faible), lui permettant ainsi d'anticiper ses commandes et d'optimiser la gestion de son stock.

2.6 Conclusion

Dans ce chapitre, nous nous sommes intéressés à l'analyse et à la conception du système PharmaVision 4.0. Nous avons commencé par une étude des besoins menée à travers un questionnaire, afin d'identifier l'existant dans la gestion des stocks pharmaceutiques et de révéler les points sensibles : des ruptures de stock fréquentes chez certains pharmaciens, l'absence quasi générale d'outils de prévision, et le fait que la plupart des répondants se déclarent favorables à une solution automatisée, confirmant qu'une solution basée sur l'intelligence artificielle constitue un véritable atout.

Nous avons ensuite présenté l'architecture générale du système PharmaVision 4.0. Afin de mieux illustrer son fonctionnement et de comprendre les interactions entre ses composants matériels et logiciels, nous avons opté pour une modélisation UML : cette approche a permis de structurer la description du système et de représenter de manière formelle ses différents comportements, sous forme d'un diagramme de cas d'utilisation et de diagrammes de séquence. L'architecture proposée sépare clairement les responsabilités : les Raspberry Pi assurent la détection et le comptage en temps réel directement sur le terrain, le serveur central garantit la persistance des données et la cohérence de l'inventaire, et le tableau de bord offre au pharmacien une interface de supervision complète. Ce dernier intègre également un module de prévision basé sur le modèle Bi-LSTM, qui exploite les ventes des trois derniers mois (M, M-1 et M-2) pour générer une prédiction du mois suivant et la sauvegarder automatiquement sous forme de fichier Excel, permettant ainsi au pharmacien d'anticiper au mieux ses réapprovisionnements.

Ce travail d'analyse et de conception était donc une étape nécessaire avant de passer à la réalisation concrète. Le chapitre suivant présentera l'implémentation physique du système ainsi que les résultats expérimentaux obtenus lors des tests.

Chapitre 3

Réalisation et résultats expérimentaux

Introduction

Ce chapitre présente l'ensemble du travail de réalisation du système PharmaVision 4.0, depuis la création des données jusqu'à la mise en place finale. Il couvre cinq grandes étapes qui s'enchaînent et forment ensemble un système complet de gestion intelligente de stock pharmaceutique.

La première étape a consisté à construire un dataset personnalisé à partir d'images réelles collectées à la Pharmacie MERZOUK de Tlemcen. Ensuite, nous avons entraîné un modèle de détection d'objets (YOLOv8) capable de reconnaître 105 médicaments différents. En parallèle, nous avons développé un module de prévision des ventes basé sur un réseau Bi-LSTM, qui anticipe les besoins futurs grâce à 41 mois d'historique de ventes réelles.

Une fois que les modèles ont été validés, nous avons déployé tout le système dans une architecture à trois niveaux. D'abord, des Raspberry Pi embarquent le modèle YOLOv8 et détectent en direct les médicaments posés sur les étagères de notre maquette. Ensuite, un serveur central se charge de récupérer et stocker toutes les données. Enfin, un tableau de bord web permet au pharmacien d'avoir une vue complète interactive de l'état du stock.

3.1 Création du dataset

La première étape de tout projet de vision par ordinateur est la constitution d'un dataset. Un dataset est une collection d'images annotées qui sera utilisée pour entraîner un modèle de détection d'objet. La qualité et la quantité de ces images déterminent les performances finales du modèle.

3.1.1 Collecte des images

Dans ce projet, nous avons eu la chance de travailler avec la Pharmacie MERZOUK à Tlemcen. Ils nous ont ouvert leurs stocks et nous ont laissé photographier les médicaments. Au total, nous avons pris 772 photos, couvrant 105 médicaments différents. Nous avons varié les éclairages, les angles de prise de vue et les dispositions sur les étagères, pour que le modèle soit robuste.

3.1.2 Annotation des données

Puis vient une étape très importante mais aussi très longue et fatigante qui est l'annotation des images. Nous avons utilisé Roboflow pour cette tâche. C'est une plateforme qui offre la capacité de rassembler, transformer, augmenter et gérer des paramètres pour pouvoir produire une base de données personnalisée [20].

Cette plateforme nous a permis de faire les trois opérations suivantes :

— **L'annotation :** elle consiste à dessiner manuellement des rectangles (*bounding boxes*) autour de chaque boîte de médicament et à attribuer des étiquettes portant leurs noms.

— **Le prétraitement et l'augmentation :** Roboflow permet d'augmenter une base de données en appliquant des transformations comme le redimensionnement, la rotation, le retournement, les variations de luminosité, etc., ce qui enrichit la diversité des données et améliore la robustesse du modèle.

— **L'export :** une fois le dataset prêt, la plateforme permet de l'exporter dans le format YOLO compatible avec la bibliothèque Ultralytics.

Au final, notre base de données compte 2008 images, réparties en 80% entraînement, 10% validation, 10% test.

3.2 Entraînement du modèle YOLO

3.2.1 Paramètres d'entraînement

L'entraînement du modèle YOLOv8 sur notre propre dataset a été réalisé sur Google Colab, en utilisant un GPU T4 doté de 16 Go de VRAM.

Les principaux paramètres de cet entraînement sont les suivants :

— **Nombre d'époques :** Une époque correspond à un passage complet de toutes les images d'entraînement dans le modèle. Nous en avons fixé 200 pour laisser le temps au modèle d'apprendre correctement.

— **Patience :** La patience est un mécanisme d'arrêt anticipé. Si le modèle ne progresse pas pendant 50 epochs, l'entraînement s'arrête tout seul.

— **Taille des images :** Toutes les images ont été redimensionnées à 640×640 pixels, car les boîtes de médicaments occupent l'ensemble de l'image, y compris les bords. Réduire davantage risquerait de couper ou d'écraser certains médicaments.

— **Batch size :** 16, ce qui correspond au nombre d'images traitées simultanément.

— **Optimiseur :** L'optimiseur est l'algorithme qui ajuste les poids internes du modèle pour réduire ses erreurs. Nous avons choisi AdamW, une version améliorée du

célèbre optimiseur Adam. Le taux d'apprentissage est fixé à 0,001, valeur recommandée pour AdamW.

— **Cos_lr :** activé (True), cela permet d'adapter le taux d'apprentissage en suivant une courbe cosinus.

— **Augmentation des données :** activée (True). YOLO génère automatiquement des variations aléatoires à chaque époque, ce qui améliore l'apprentissage.

— **Pré-entraînement :** nous avons utilisé les poids de YOLOv8 déjà entraînés sur COCO, un dataset qui contient plus de 330000 images et 80 classes d'objets courants, puis nous avons continué l'entraînement sur notre propre base de données de médicaments. Cette étape s'appelle le *finetuning*.

3.2.2 Évaluation des performances

À la fin de l'entraînement, nous avons récupéré plusieurs métriques pour juger la qualité du modèle :

— **Précision (P) :** parmi toutes les boîtes détectées par le modèle, combien sont vraiment correctes.

— **Rappel (R) :** parmi toutes les boîtes réelles présentes dans les images, combien le modèle a trouvées.

— **mAP50 :** cette métrique mesure la performance du modèle en considérant qu'une détection est correcte si le chevauchement entre la boîte prédite et la vraie boîte est d'au moins 50%. On calcule ensuite la moyenne de ce paramètre pour toutes les classes.

— **mAP50-95 :** on calcule la même moyenne, mais pour des seuils de chevauchement allant de 50% jusqu'à 95%. Cette métrique est beaucoup plus exigeante car elle oblige le modèle à être très précis.

Nous avons aussi regardé les courbes de pertes (box_loss, cls_loss, dfl_loss) et la matrice de confusion pour voir si le modèle confondait les médicaments entre eux.

3.2.3 Résultats expérimentaux

Une fois l'entraînement terminé, nous avons évalué YOLOv8 sur notre jeu de validation. Voici les principales métriques obtenues.

mAP La mAP50 atteint 0,817 (81,7%). Cela signifie que, dans huit cas sur dix, le modèle détecte correctement un médicament avec une boîte qui recouvre au moins la moitié de la vraie boîte.

La mAP50-95 est de 0,522. Cette valeur est plus basse, ce qui est normal car elle exige que la boîte prédite soit très bien ajustée.

Précision La précision mesurée est de 0,796 (environ 79,6%). Quand le modèle annonce une détection, il a environ quatre chances sur cinq d'avoir raison.

Rappel Le rappel est de 0,765 (environ 76,5%). Le modèle ne manque qu'un quart des boîtes de médicaments réellement présentes dans les images.

Matrice de confusion Aucune confusion entre les classes : toutes les valeurs hors diagonale sont à zéro. Détections correctes par classe (sélection) :

Médicament	**Détections correctes**
Dolyc	46
Gripex	30
Paralgan	17
Panadol	17
Efferalgan	16
Vitamine C	12
Multi vitamine	11
Xydol 200	7
Panadol rouge	0
Ibuprofène 600	0

Les deux dernières classes n'ont aucune détection correcte car elles étaient moins présentes dans les images de validation. Malgré cela, le modèle ne les confond jamais avec d'autres médicaments.

3.3 Module de prévision de la demande

3.3.1 Collecte des données

Les données utilisées proviennent des registres mensuels de sorties de médicaments de la pharmacie MERZOUK à Tlemcen. Nous avons au total 41 fichiers au format XLS couvrant 41 mois consécutifs, de janvier 2023 à mai 2026.

TABLE 3.1 – Contenu de l'un des fichiers Excel

	A	B	C	D
1	général	sortie		
7	aspirine c upsa 330mg/200mg comp. effervescent b/20	5,00		
8	biofenac 100mg comp.pell. ... lp bte/20	6,00		
9	biofenac 100mg suppositoire b/10	6,00		
10	biofenac fast 50 mg pdr p/sol buv sachets bte/10	2,00		
11	biofenac lp 75 mg comp.pell. ... lp bte/30	5,00		
12	celebrex 200 mg gelules bte/30	3,00		
13	celebrex 200mg gelules bte/15	0,00		
14	celetasone 2mg comp. dispersible bte/20	7,00		
15	clamoxyl 1g comp. dispersible b/14	50,00		
16	clofenal 100mg suppositoire b/10	25,00		
17	cycladol 191,2 mg equiv 20 mg piroxicam comprime sec. b/14	3,00		
18	desacrovis 0,5mg comprime b/30	3,00		
19	divido 75mg gles/double lib bte/20	9,00		
20	doliprane 1000mg comprime bte/8	53,00		
21	doliprane 100mg suppositoire b/10	5,00		
22	doliprane 150mg suppositoire b/10	7,00		
23	doliprane 200mg suppositoire b/10	12,00		
24	doliprane 300mg pdr.p/sol.buv(sachet) b/12	0,00		
25	doliprane 300mg suppositoire b/10	18,00		
26	dolyc 1000mg comprime bte/10	222,00		
27	dolyc 500mg comprime b/20	47,00		
28	domperidone 10mg comprime b/30	0,00		
29	efferalgan 1000mg comp. effervescent b/08	17,00		
30	efferalgan pediatrique 3% solution buvable fl/150ml	36,00		
31	etofan 90mg comp.pellicule b/30	0,00		

3.3.2 Prétraitement des données

Lecture et normalisation des noms Un script dans Google Colab parcourt automatiquement les 41 fichiers XLS. Pour chacun, il lit le nom du fichier (par exemple aout_23.xls), extrait le mois et l'année, et normalise les noms de médicaments (minuscules, suppression des espaces superflus).

Construction du tableau pivot Toutes les données sont rassemblées dans un seul tableau croisé où les lignes représentent les mois (ordonnés chronologiquement) et les colonnes représentent les médicaments. Si un médicament n'a pas été vendu un mois donné, la valeur est mise à zéro.

Normalisation Pour corriger les différences d'échelle entre médicaments, on applique une normalisation *MinMaxScaler* :

$$x_{\text{norm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{3.1}$$

Ainsi, toutes les séries sont ramenées dans l'intervalle $[0, 1]$, et le réseau peut apprendre leurs dynamiques respectives sans qu'une écrase l'autre.

Construction des séquences d'entraînement Les données sont transformées en séquences glissantes de longueur SEQ_LEN = 3 mois. À chaque pas de temps, deux informations sont fournies au modèle : la quantité normalisée du médicament, et le numéro du mois divisé par 12 (indicateur saisonnier).

3.3.3 Architecture du modèle

Le modèle utilisé est un réseau LSTM bidirectionnel (Bi-LSTM) implémenté sous PyTorch. L'architecture est la suivante :

```python
class BiLSTM(nn.Module):
    def __init__(self):
        super().__init__()
        self.lstm = nn.LSTM(
            input_size    = 2,
            hidden_size   = 32,
            num_layers    = 1,
            batch_first   = True,
            bidirectional = True,
            dropout       = 0.0
        )
        self.fc = nn.Sequential(
            nn.Linear(64, 16),
            nn.ReLU(),
            nn.Linear(16, 1)
        )
    def forward(self, x):
        out, _ = self.lstm(x)
        return self.fc(out[:, -1, :]).squeeze(1)
```

FIGURE 3.1 – Architecture du modèle BI-LSTM.

TABLE 3.2 – Architecture du modèle Bi-LSTM

Couche	**Configuration**	**Rôle**
Entrée	input_size=2	À chaque pas de temps, le modèle reçoit deux chiffres : la quantité normalisée et le mois divisé par 12.
Bi-LSTM	hidden_size=32, bidirectional=True	Cœur du modèle. Lit la séquence de 3 mois dans les deux sens pour détecter tendances et cycles.
Sortie LSTM	64 neurones (32×2)	Concaténation des sorties des deux directions.
Couche dense 1	Linear(64, 16) + ReLU	Compresse les 64 valeurs en 16 en conservant les informations les plus importantes.
Couche dense 2	Linear(16, 1)	Produit la prédiction finale (quantité normalisée pour le mois suivant).

3.3.4 Entraînement du modèle

Les paramètres de l'entraînement du modèle Bi-LSTM sont les suivants :

— **SEQ_LEN** : fixé à 3, ce paramètre correspond au nombre de mois consécutifs fournis en entrée du modèle.

— **Epochs** : fixé à 200, il représente le nombre de passages complets sur l'ensemble des données d'entraînement.

— **LR (taux d'apprentissage)** : fixé à 0,001, il détermine l'amplitude des mises à jour des poids du réseau.

— **Optimiseur** : Adam est utilisé ; il ajuste automatiquement le taux d'apprentissage au cours de l'entraînement pour améliorer la convergence.

— **Fonction de perte** : MSELoss est employée pour mesurer l'écart entre la valeur prédite par le modèle et la valeur réelle.

— **Division train/test** : les données sont réparties tel que, les 80% premiers mois servent à l'entraînement, et les 20% restants aux tests.

3.3.5 Résultats et interprétations

Nous présentons ci-dessous les résultats de l'entrainement (Figures 3.2 et 3.3).

```
  RMSE : 6.3420
  MAE  : 3.2931
  R²   : 0.9239
  MAPE : 58.19%
MAPE réel (valeurs ≥ 5) : 31.63%
```

FIGURE 3.2 – Résultat de l'entrainement.

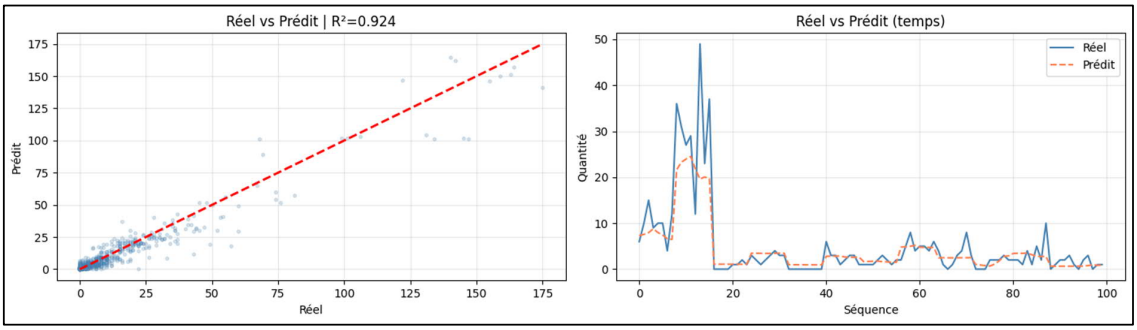


FIGURE 3.3 – Courbes d'évaluation du modèle Bi-LSTM

— **R^2 = 0,9239 :** Le modèle explique 92% de la variance totale des consommations, ce qui est un excellent résultat.

— **MAE = 3,29 boîtes :** En moyenne, le modèle se trompe de 3 boîtes, ce qui est une erreur faible.

— **RMSE = 6,34 boîtes :** La RMSE plus élevée que la MAE indique quelques erreurs importantes sur les médicaments à forte consommation.

— **MAPE brut = 58,19% :** Cette valeur élevée s'explique par les médicaments à très faible consommation (0 ou 1 boîte par mois) qui produisent des pourcentages d'erreur très élevés.

— **MAPE réel = 31,63% (valeurs $\geq$ 5) :** Sur les médicaments qui se vendent fréquemment, le modèle se trompe en moyenne de 32%, ce qui est acceptable.

3.4 Détection en temps réel

3.4.1 Déploiement sur Raspberry Pi

Chaque Raspberry Pi exécute un programme Python nommé main.py qui assure quatre fonctions essentielles : capturer les images de la caméra, les faire analyser par

le modèle YOLOv8, produire un inventaire en temps réel et exposer les résultats via un serveur web.

3.4.2 Paramètres de configuration

— CONF = 0.35 : seuil de confiance. Toute détection avec une probabilité inférieure à 35% est ignorée.

— NMS = 0.45 : seuil de *Non-Maximum Suppression*. Lorsque plusieurs rectangles se chevauchent à plus de 45%, seul celui ayant la plus haute confiance est conservé.

— SIZE = 640 : toutes les images capturées sont redimensionnées à 640×640 pixels avant d'être soumises au modèle.

3.4.3 Chargement du modèle NCNN

Le modèle YOLOv8 est converti au format NCNN pour être exécuté efficacement sur le processeur du Raspberry Pi. Deux fichiers sont chargés :

— model.ncnn.param : décrit l'architecture du réseau.

— model.ncnn.bin : contient les poids du réseau.

3.4.4 Initialisation de la caméra

La bibliothèque Picamera2 configure le module caméra pour capturer des images en 1280×720 au format BGR. La commande time.sleep(1) introduit une pause d'une seconde pour que la caméra stabilise ses paramètres. La configuration ScalerCrop-Maximum garantit que la caméra utilise toute la surface du capteur, ce qui maximise le champ de vision.

3.4.5 Boucle de détection en temps réel

La boucle detection_loop() est le cœur opérationnel du système. Elle s'exécute en parallèle avec le serveur Flask et tourne de façon infinie. À chaque itération, elle effectue les étapes suivantes :

TABLE 3.3 – Étapes de la boucle de détection

Étape	Action	Résultat
1	Capture d'une image	Tableau NumPy 1280×720
2	Prétraitement	Image redimensionnée à 640×640 pixels
3	Normalisation	Pixels divisés par 255, valeurs dans $[0, 1]$
4	Inférence IA	Coordonnées des boîtes englobantes et probabilités par classe
5	Post-traitement (NMS)	Sélection de la classe la plus probable ($> 0{,}35$) et suppression des détections redondantes via cv2.dnn.NMSBoxes()
6	Affichage	Dessin des rectangles sur l'image et mise à jour du dictionnaire inventory
7	Serveur Flask	Exposition des données sur le réseau local

3.4.6 Le Serveur Flask et ses Routes

Flask est un framework web léger pour Python permettant de créer un serveur HTTP. Dans le code de détection, Flask met à disposition deux routes :

— **/video_feed :** fournit un flux vidéo en temps réel via le protocole MJPEG streaming.

— **/inventory :** retourne l'inventaire courant au format JSON, associant à chaque médicament sa quantité détectée.

3.4.7 Architecture multi-thread

Le programme main.py utilise une architecture multi-thread :

— **Thread de détection :** tourne en arrière-plan, capture les images, les analyse avec YOLO et met à jour l'inventaire.

— **Thread Flask :** lit les variables globales mises à jour et les transmet aux clients.

3.5 Le Serveur Central

Les deux Raspberry Pi produisent chacun un inventaire basé sur ce qui est détecté par leur caméra. Le serveur central, qui tourne sur le PC du pharmacien, assure trois fonctions :

1. Il interroge régulièrement les deux Raspberry Pi pour récupérer leurs inventaires.
2. Il fusionne les deux inventaires, détecte les variations et enregistre tout dans une base de données SQLite.

3. Il fournit une API REST pour que le tableau de bord puisse récupérer les données.

3.5.1 La base de données SQLite

La base de données pharma.db contient trois tables :

— **Table dernier_inventaire :** mise à jour à chaque itération avec le dernier état du stock.

— **Table historique :** enregistre toutes les variations de stock (date, médicament, quantités avant et après).

— **Table alertes :** contient uniquement les médicaments en rupture ou en stock faible.

3.5.2 Les bibliothèques utilisées

TABLE 3.4 – Bibliothèques Python du serveur central

Bibliothèque	**Rôle**
flask	Création du serveur HTTP et des routes API
flask_cors	Autorise les requêtes cross-origin entre le tableau de bord et le serveur
sqlite3	Interface entre le serveur et le fichier pharma.db
requests	Récupère l'inventaire depuis les Raspberry Pi via HTTP
threading	Permet d'exécuter la collecte et le serveur Flask en parallèle
datetime	Enregistre la date et l'heure exactes de chaque événement

3.5.3 La boucle de collecte

La fonction boucle_collecte() tourne dans un thread séparé avec une pause de 3 secondes entre chaque cycle. À chaque itération :

— Elle envoie une requête HTTP GET à l'adresse /inventory de chaque Raspberry Pi (timeout de 3 secondes).

— Elle fusionne les deux inventaires reçus.

— Elle compare l'inventaire fusionné avec celui du cycle précédent et insère une ligne dans la table historique pour chaque variation.

— Elle écrase la table dernier_inventaire avec l'inventaire actuel.

— Elle reconstruit la table alertes en appliquant deux règles, une rupture est enregistrée si la quantité détectée est égale à 0, une alerte est enregistrée si la quantité

est supérieure à 0 mais inférieure ou égale au seuil appelé seuil AI calculé à partir de la prediction bi-lstm :

$$\text{seuil} = \left(\frac{\text{quantité_prédite}}{30}\right).(10+3)$$

où :

— Consommation journalière = $\frac{\text{quantité_prédite}}{30}$

— Délai de livraison = 10 jours

— Stock de sécurité = 3 jours

Contrairement à l'approche traditionnelle où le pharmacien fixe manuellement un seuil d'alerte selon son expérience personnelle, ce qui reste subjectif et approximatif, le seuil utilisé dans PharmaVision 4.0 est calculé automatiquement par le système à partir des prédictions du modèle Bi-LSTM. Il tient compte à la fois de la consommation estimée pour le mois suivant et des contraintes logistiques comme le délai de livraison et stock de sécurité, ce qui le rend objectif, adaptatif et propre à chaque médicament.

— Elle attend 3 secondes et recommence.

3.5.4 Les routes API REST

TABLE 3.5 – Routes API REST du serveur central

Route	Source	Contenu
/api/inventaire	Variables globales	Inventaire temps réel + statut des Raspberry Pi
/api/dernier_inventaire	Table BDD	Dernier état de stock connu
/api/historique	Table BDD	Toutes les variations de stock
/api/alertes	Table BDD	Médicaments en rupture ou stock faible

3.5.5 Architecture multi-thread

Comme pour main.py, le serveur central utilise deux threads :

— **Thread de collecte :** exécute boucle_collecte() en arrière-plan.

— **Thread principal (Flask) :** écoute les requêtes HTTP et répond au tableau de bord.

3.5.6 Séquence de démarrage

Quand on lance le serveur, les étapes suivantes se déroulent dans l'ordre :

1. **Préparation de la base de données :** vérification et création des tables si nécessaire.
2. **Démarrage de la collecte :** rechargement du dernier inventaire connu.
3. **Collecte continue :** interrogation des Raspberry Pi toutes les 3 secondes.
4. **Mise en route de l'API :** activation du serveur web Flask.
5. **Visualisation :** le tableau de bord interroge les routes API pour afficher les données.

3.6 Tableau de bord (Dashboard)

3.6.1 Structure de l'interface (HTML/CSS/JS)

L'interface du tableau de bord est développée en HTML/CSS/JS pur, sans framework externe. Elle s'articule autour de deux zones principales : une barre de navigation latérale fixe à gauche et une zone de contenu principale à droite.

— **La barre latérale :** contient le logo du système et les liens de navigation organisés en deux groupes (« Navigation » et « Analyse ») : Accueil, Caméras, Inventaire, Alertes, Historique et Exportation.

— **L'en-tête :** affiche un voyant indiquant si le serveur est connecté et une horloge mise à jour chaque seconde.

— **La section Accueil :** présente quatre cartes de statistiques (médicaments détectés, types distincts, caméras actives, alertes) et un tableau résumant l'état des deux racks (Figure 3.4).

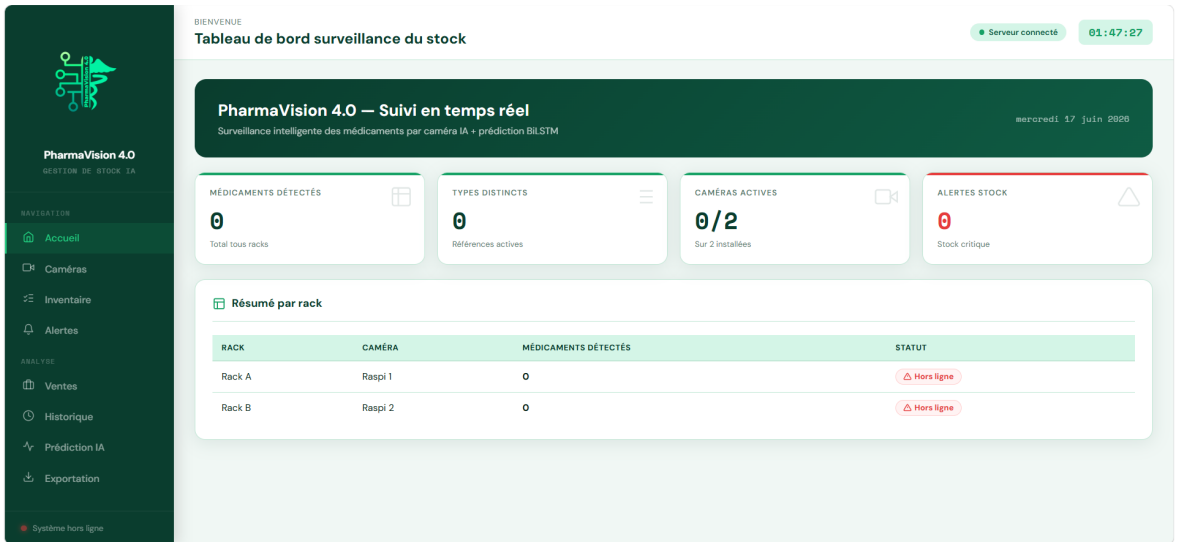


FIGURE 3.4 – Tableau de bord section Accueil

— **La section Caméras :** affiche les flux vidéo en direct des deux Raspberry Pi avec un tableau des médicaments détectés par chaque caméra.

— **La section Inventaire :** propose un tableau complet des 105 médicaments avec une barre de recherche et des filtres par statut.

— **La section Prédiction IA** : permet de lancer l'entraînement du modèle BiLSTM et de consulter les prévisions de demande générées pour le mois suivant (Figure 3.5).

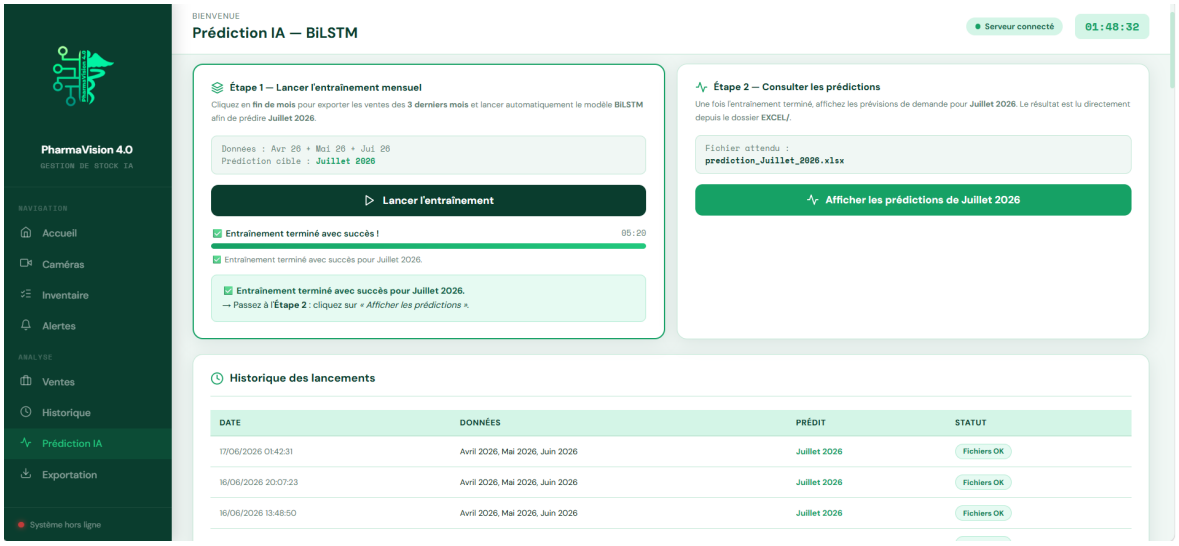


FIGURE 3.5 – Tableau de bord section Prédiction IA

3.6.2 Récupération des données depuis le serveur et les deux Raspberry Pi

Dès que le système est lancé, le tableau de bord interroge le serveur toutes les 3 secondes pour récupérer les données de détection des deux Raspberry Pi. Les flux vidéo en direct s'affichent directement dans l'interface avec un badge « LIVE » qui devient rouge dès qu'une caméra n'est plus accessible.

3.6.3 Fusion des inventaires et gestion des alertes

Les données des deux caméras sont automatiquement fusionnées pour produire un inventaire global. Chaque médicament reçoit un statut visuel selon sa quantité comparée au seuil AI calculé à partir des prédictions Bi-LSTM du mois suivant. Dès qu'un médicament passe en dessous du seuil, une alerte se déclenche.

3.6.4 Historique des variations et exports

Chaque fois qu'une quantité change, le système enregistre la date et l'heure exactes, l'ancienne valeur, la nouvelle, et la variation calculée. Les augmentations sont affichées en vert, les baisses en rouge. L'utilisateur peut exporter les données au format CSV ou TXT directement depuis le navigateur.

3.7 Discussion générale des résultats

Le système PharmaVision 4.0 repose sur trois composantes complémentaires : un modèle de détection visuelle YOLOv8, un module de prévision de la demande Bi-LSTM, et une architecture distribuée de surveillance en temps réel.

Côté détection, YOLOv8 atteint une mAP50 de 81,7% et une précision de 79,6%. Le rappel de 76,5% confirme que la grande majorité des produits sont bien détectés. Le résultat le plus encourageant est l'absence de toute confusion entre les 105 médicaments.

Pour la prévision des demandes, le modèle Bi-LSTM obtient un R^2 de 0,92, et une MAE de 3,29 boîtes. Le MAPE réel de 31,63% est tout à fait acceptable compte tenu du fait que le modèle ne dispose que de 41 mois de données.

L'architecture globale du système fonctionne bien : la communication entre les Raspberry Pi, le serveur central et le tableau de bord est fluide, les données sont bien stockées dans SQLite, et le multi-thread permet à la collecte et à l'affichage de tourner en parallèle.

Malgré ces bons résultats, plusieurs limites ont été relevées :

— Le jeu de données n'est pas totalement équilibré, ce qui explique que certains médicaments (Panadol rouge, Ibuprofène 600) ne sont pas bien détectés.

— L'annotation manuelle d'environ 700 images, avec 10 à 20 boîtes par image, était extrêmement longue dans le cadre d'un PFE de 3 mois.

— L'historique de 41 mois est assez court pour entraîner un réseau de neurones.

— Le Raspberry Pi 3 est relativement lent, ce qui limite la synchronisation en temps réel et oblige à placer la caméra très près des étagères.

En perspective, migrer vers un Raspberry Pi 5, élargir le dataset à l'ensemble des références de la pharmacie, et disposer d'un historique plus long permettraient d'améliorer significativement les performances du système.

3.8 Conclusion

Ce chapitre a détaillé toute la conception et la réalisation du système PharmaVision 4.0. Les résultats expérimentaux sont globalement satisfaisants. Le modèle YOLOv8 atteint une mAP50 de 81,7% sans aucune confusion entre les 105 médicaments. Le Bi-LSTM obtient un R^2 de 0,92, confirmant sa capacité à modéliser correctement les tendances de consommation mensuelle.

L'architecture globale, articulée autour de Raspberry Pi, d'un serveur central avec Flask et SQLite, et d'un tableau de bord web, s'est avérée fonctionnelle, légère et robuste. Grâce au multi-thread, le système continue de tourner même si un des Raspberry Pi tombe en panne.

La contrainte de temps représente la limite principale de ce projet. Si l'on disposait de davantage de temps, on pourrait nettement améliorer le système, que ce soit en enrichissant le base de donnée des images ou en s'appuyant sur un historique de ventes plus étoffé.

Conclusion générale

Au début de ce travail, nous sommes partis d'un constat simple mais préoccupant : une erreur de stock en pharmacie n'est pas seulement une perte financière, c'est avant tout un risque pour les patients. Un médicament introuvable au bon moment peut retarder un traitement, aggraver un état de santé, voire mettre une vie en danger. Pourtant, les outils utilisés aujourd'hui dans la plupart des officines sont l'inventaire manuel ou logiciels à codes-barres qui restent insuffisants face à cette réalité. Ces systèmes dépendent de l'humain à chaque étape, ils ne peuvent pas observer directement ce qui se passe dans les rayons. Le questionnaire que nous avons mené auprès des pharmaciens en Algérie l'a confirmé, tous subissent des ruptures de stock, plus de 85% n'utilisent aucun outil de prévision, et une grande majorité se déclare favorable à une solution basée sur l'intelligence artificielle. Le premier chapitre nous a permis de poser des bases théoriques solides. Nous avons passé en revue les méthodes classiques de gestion des stocks et leurs limites, exploré les principes de l'Industrie 4.0 et leur application dans le domaine pharmaceutique à travers la Pharma 4.0, puis étudié les technologies au cœur de notre système qui sont la vision par ordinateur avec YOLOv8, le Raspberry Pi comme plateforme embarquée, et les réseaux de neurones Bi-LSTM pour la prévision de la demande. Ce chapitre nous a montré que les technologies nécessaires existent déjà et ont fait leurs preuves dans beaucoup de domaines, mais le véritable défi consistait à les assembler pour concevoir un système complet, adapté aux contraintes réelles d'une pharmacie. Le deuxième chapitre a transformé cette vision en une architecture concrète. Deux Raspberry Pi équipés de caméras surveille en permanence les racks, un serveur central récupère et fusionne les données des deux zones de stockage, un tableau de bord web offre au pharmacien une vue claire et instantanée de son stock, avec des alertes automatiques, un historique des mouvements et la possibilité d'exporter les données. La modélisation UML a permis de structurer formellement les interactions entre les composants, de valider la logique du système et de clarifier les responsabilités de chaque acteur. Le troisième chapitre a présenté la mise en œuvre concrète du système ainsi que les résultats expérimentaux obtenus. Concernant la détection de boites de médicaments, le modèle YOLOv8 entraîné sur plus de 2000 images collectées à la Pharmacie MERZOUK de Tlemcen, a atteint une mAP50 de 81,7%, une précision de 79,6% et un rappel de 76,5%, confirmant que la grande majorité des produits sont correctement détectés. Le résultat le plus remarquable reste l'absence totale de confusion entre les 105 médicaments. C'est une propriété capitale dans un contexte pharmaceutique, où une confusion entre deux références peut avoir des conséquences directes sur la gestion du stock. Coté prévision de la demande, le modèle

Bi-LSTM, entraîné sur 41 mois d'historique de ventes réelles, a obtenu un coefficient de détermination R^2 de 0,92, ce qui signifie qu'il explique 92% de la variance totale des consommations observées. Une erreur absolue moyenne de seulement 3,29 boîtes par mois confirme la fiabilité des prédictions sur les médicaments à vente régulière. Le MAPE réel de 31,63%, calculé uniquement sur les médicaments avec des volumes significatifs, est tout à fait acceptable pour un modèle disposant d'un historique aussi limité. Ces résultats démontrent que le Bi-LSTM est capable de capturer les tendances saisonnières et les cycles de consommation, permettant ainsi d'anticiper les ruptures de stock plutôt que de les subir. L'architecture distribuée du système s'est avérée fonctionnelle, légère et robuste. La communication entre les Raspberry Pi, le serveur central et le tableau de bord est fluide et fiable. L'utilisation du multithread garantit que la collecte des données et l'affichage continuent de fonctionner en parallèle, même en cas de défaillance de l'un des Raspberry Pi. La base de données SQLite assure la persistance de l'historique et la reprise correcte du système après un redémarrage, sans perte d'information. Comme perspectives, plusieurs axes d'amélioration peuvent être envisagés. L'enrichissement de la base de données d'images par l'intégration de l'ensemble du catalogue de médicaments de la pharmacie, notamment les références les moins représentées, permettrait d'améliorer davantage les performances du modèle YOLOv8 et sa capacité de généralisation. De même, la disponibilité d'un historique de ventes plus long offrirait au modèle Bi-LSTM davantage d'informations temporelles pour mieux capturer les cycles saisonniers, les variations annuelles et les comportements exceptionnels de consommation. Sur le plan matériel, l'utilisation de plateformes embarquées plus puissantes, telles que le Raspberry Pi 5 ou les cartes NVIDIA Jetson, permettrait d'augmenter la fréquence d'inférence, d'améliorer la qualité des images traitées et d'obtenir des performances en temps réel encore plus élevées. L'intégration de techniques d'apprentissage continu pourrait également permettre au système de s'adapter automatiquement à l'apparition de nouveaux médicaments sans nécessiter un réentraînement complet du modèle. PharmaVision 4.0 pourrait être étendu vers un système intelligent de gestion pharmaceutique à l'échelle d'un réseau d'officines, capable de partager les informations de stock entre plusieurs pharmacies, de coordonner les approvisionnements et d'optimiser la disponibilité des médicaments à l'échelle régionale. L'intégration de technologies complémentaires telles que l'Internet des Objets (IoT), le cloud computing et les modèles avancés d'intelligence artificielle ouvrirait la voie à une gestion prédictive et autonome des stocks pharmaceutiques. Ce travail démontre qu'il est possible de concevoir une solution intelligente, fiable et économiquement accessible pour moderniser la gestion des stocks en pharmacie. En combinant vision par ordinateur, apprentissage profond et systèmes embarqués, PharmaVision 4.0 constitue une contribution concrète à la transformation numérique du secteur pharmaceutique et s'inscrit pleinement dans la dynamique de la Pharma 4.0. Au-delà des performances

obtenues, la véritable valeur de ce système réside dans sa capacité à aider les pharmaciens à prendre de meilleures décisions, à réduire les risques de rupture de stock et à contribuer à une meilleure qualité de prise en charge des patients.

Annexe A

Questionnaire destiné aux professionnels de la pharmacie

Objectif : Identifier les besoins liés à la gestion des stocks pharmaceutiques et évaluer l'intérêt d'une solution intelligente basée sur la vision par ordinateur et l'intelligence artificielle.

Informations générales

1. Quelle est votre fonction ?

☐ Pharmacien

☐ Gestionnaire de stock

☐ Vendeur en pharmacie

☐ Autre : .....

2. Nombre d'années d'expérience :

☐ Moins de 2 ans

☐ 2 à 5 ans

☐ 5 à 10 ans

☐ Plus de 10 ans

Gestion actuelle des stocks

3. Rencontrez-vous des ruptures de stock ?

☐ Jamais

☐ Rarement

☐ Parfois

☐ Souvent

4. Comment gérez-vous actuellement votre stock ?

☐ Manuellement

☐ Logiciel de gestion

☐ Méthode mixte

5. Les erreurs d'inventaire sont-elles fréquentes ?

☐ Oui

☐ Non

Prévision de la demande

6. La prévision de la demande est-elle importante pour votre activité ?

☐ Pas importante

☐ Peu importante

☐ Importante

☐ Très importante

7. Utilisez-vous actuellement des outils de prévision ?

☐ Oui

☐ Non

Évaluation de la solution proposée

8. Une détection automatique des médicaments par caméra pourrait-elle être utile ?

☐ Pas utile

☐ Peu utile

☐ Utile

☐ Très utile

9. Un système d'alerte automatique des ruptures de stock serait-il intéressant ?

☐ Oui

☐ Non

10. Un tableau de bord en temps réel faciliterait-il votre travail ?

☐ Oui

☐ Non

11. Quels indicateurs aimeriez-vous suivre ? *(Plusieurs choix possibles)*

☐ Quantité en stock

☐ Médicaments à faible stock

☐ Prévision de la demande

☐ Rotation du stock

☐ Historique des ventes

☐ Autre : .....

12. Seriez-vous favorable à l'utilisation d'une solution basée sur l'intelligence artificielle pour la gestion des stocks ?

☐ Très défavorable

☐ Défavorable

☐ Favorable

☐ Très favorable

Merci pour votre participation

Références bibliographiques

[1] Yousfi, R., & Mouri, M. O. (2023). *Gestion des stocks des produits pharmaceutiques (médicaments et dispositifs médicaux) : Cas de CHU de Bejaïa* [Mémoire de master, Université Abderrahmane Mira de Bejaïa].

[2] Kim, D. (s. d.). *Comment l'industrie 4.0 révolutionne les opérations des entrepôts industriels*. AutoStore. https://www.autostoresystem.com/fr/insights/how-industry-4-0-is-revolutionizing-industrial-warehousing-operations

[3] RS Components. (2024, 11 septembre). *L'industrie 4.0 dans le secteur pharmaceutique*. RS Components. https://fr.rs-online.com/web/content/blog-discovery/maintenance/industrie-4-0-pharmaceutique

[4] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You only look once : Unified, real-time object detection*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779–788.

[5] Nayyar, A., & Puri, V. (2015). *Raspberry Pi — A small, powerful, cost effective and efficient form factor computer : A review*. International Journal of Advanced Research in Computer Science and Software Engineering, 5(12), 720–737.

[6] Tecnoloblog. (2025, 14 septembre). *Applications industrielles du Raspberry Pi : guide complet et cas d'utilisation*. Tecnoloblog. https://tecnoloblog.com/fr/Utilisations-du-Raspberry-Pi-dans-l'industrie/

[7] Lagnoux, A. (s.d.). *Séries chronologiques*. ISMAG, Master 1 - MI00141X, Université de Toulouse Le Mirail.

[8] Gzl, R. (2023, 14 septembre). *Algorithmes de régression en machine learning*. Scribd. Consulté le 6 mai 2026, à l'adresse https://www.scribd.com/document/671317823/regression

[9] Salman, H. A., Kalakech, A., & Steiti, A. (2024). *Random Forest Algorithm Overview*. Babylonian Journal of Machine Learning, Vol. 2024, pp. 69–79. https://doi.org/10.58496/BJML/2024/007. Consulté le 6 mai 2026.

[10] Chen, T., & Guestrin, C. (2016). *XGBoost : A scalable tree boosting system*. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794. https://doi.org/10.1145/2939672.2939785

[11] MathWorks. (s.d.). *Réseau de neurones*. MathWorks. Consulté le 6 mai 2026, à l'adresse https://fr.mathworks.com/discovery/neural-network.html

[12] Hochreiter, S., & Schmidhuber, J. (1997). *Long short-term memory*. Neural Computation, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[13] Abbas, J., Hu, Z., Kanwal, S., Ahmad, A., Almogren, A., & Altameem, A. (2024). *Bi-LSTM-Based Model for Classifying Software Requirements*. Preprints.org. https://doi.org/10.20944/preprints202410.2129.v1

[14] dynle. (2023).*Raspberry Pi Inventory Management System* [Logiciel]. GitHub. https://github.com/dynle/raspi-inventory-management-system

[15] Wu, W.-S., & Lu, Z.-M. (2022). *A Real-Time Cup-Detection Method Based on YOLOv3 for Inventory Management*. Sensors, 22(18), 6956. https://doi.org/10.3390/s22186956

[16] Ni, S. F., Peng, Y., & Liu, Z. J. (2022). *Logistics Demand Forecast of Fresh Food E-Commerce Based on Bi-LSTM Model*. Journal of Computer and Communications, 10, 51-65. https://doi.org/10.4236/jcc.2022.109004

[17] Siami-Namini, S., Tavakoli, N., & Siami Namin, A. (2019). *A Comparative Analysis of Forecasting Financial Time Series Using ARIMA, LSTM, and BiLSTM*. arXiv :1911.09512.

[18] Raspberry Pi Ltd. (n.d.). *Raspberry Pi 3 Model B*, Consulté le 4 juin 2026. https://www.raspberrypi.com/products/raspberry-pi-3-model-b/

[19] Raspberry Pi Ltd. (2025). *The Picamera2 Library – A libcamera-based Python library for Raspberry Pi cameras* (Version 2.0), Consulté le 4 juin 2026.

[20] Ultralytics. (s.d.). *Intégration de Roboflow avec YOLO*. Consulté sur https://docs.ultralytics.com/fr/integrations/roboflow

Resumé

Dans ce mémoire, nous nous sommes intéressés à la conception et à la réalisation d'un système intelligent nommé PharmaVision 4.0, basé sur la vision par ordinateur, le Deep Learning et une architecture embarquée. Ce système permet la supervision et la prédiction des stocks d'une pharmacie en temps réel. Ce travail répond aux besoins identifiés à travers une enquête menée auprès des pharmaciens en Algérie. Les résultats de cette enquête ont mis en évidence plusieurs limites des solutions actuellement utilisées, notamment la saisie manuelle des données, l'utilisation de logiciels de gestion classiques ne disposant pas de fonctionnalités intelligentes, ainsi que l'absence d'outils de prédiction de la demande et de détection précoce des risques de rupture de stock. Face à ces limites, nous avons proposé une solution reposant sur deux Raspberry Pi équipés de caméras et intégrant un modèle YOLOv8 entraîné sur 105 types de médicaments réels pour la détection et le suivi automatique des stocks. Cette solution est couplée à un modèle Bi-LSTM capable d'anticiper la demande future afin d'améliorer la gestion des approvisionnements. L'ensemble des informations est centralisé et restitué au pharmacien à travers un tableau de bord web interactif permettant une supervision intelligente et une prise de décision rapide.

Mots-clés : Solution intelligente, Détection d'objets, Prédiction de la demande, YOLOv8, Bi-LSTM, Raspberry Pi, Tableau de bord Web, Gestion de stock pharmaceutique.

Abstract

In this thesis, we focused on the design and implementation of an intelligent system named PharmaVision 4.0, based on computer vision, deep learning and an embedded architecture. This system enables real-time supervision and prediction of pharmaceutical stock in a pharmacy. This work responds to needs identified through a survey conducted among pharmacists in Algeria. The results of this survey highlighted several limitations of currently used solutions, including manual data entry, the use of classical management software lacking intelligent features, and the absence of demand forecasting tools and early detection of stock shortage risks. To address these limitations, we proposed a solution based on two Raspberry Pi units equipped with cameras and integrating a YOLOv8 model trained on 105 real medication types for automatic stock detection and tracking. This solution is coupled with a Bi-LSTM model capable of anticipating future demand in order to improve supply management. All information is centralized and presented to the pharmacist through an interactive web dashboard enabling intelligent supervision and rapid decision-making.

Keywords : Intelligent solution, Object detection, Demand forecasting, YOLOv8, Bi-LSTM, Raspberry Pi, Web dashboard, Pharmaceutical inventory management.

مـلـخص

في هذه المذكرة، اهتممنا بتصميم وإنجاز نظام ذكي يُسمى PharmaVision 4.0، يعتمد على الرؤية الحاسوبية والتعلم العميق والمعمارية المدمجة. يتيح هذا النظام مراقبة مخزون الأدوية والتنبؤ به في الصيدلية بشكل آني.

يستجيب هذا العمل للاحتياجات المحددة عبر استبيان أُجري على صيادلة في الجزائر، كشفت نتائجه عن قيود عدة في الحلول المستخدمة حاليًا، أبرزها الاعتماد على الإدخال اليدوي للبيانات، واستخدام برامج تسيير تقليدية تفتقر إلى الوظائف الذكية، وغياب أدوات التنبؤ بالطلب والكشف المبكر عن نفاد المخزون.

للتغلب على هذه القيود، اقترحنا حلًا يرتكز على جهازي Raspberry Pi مزودين بكاميرات، يدمجان نموذج YOLOv8 المدرَّب على 105 أنواع من الأدوية الحقيقية للكشف التلقائي عن المخزون وتتبعه، إلى جانب نموذج Bi-LSTM لاستشراف الطلب المستقبلي وتحسين إدارة التوين.

وتُعرض جميع المعلومات على الصيدلاني من خلال لوحة قيادة تفاعلية تتيح له مراقبة ذكية واتخاذ قرارات سريعة.

كلمات مفتاحية : حل ذكي، كشف الأجسام، التنبؤ بالطلب، لوحة القيادة الويب، إدارة المخزون الصيدلانيRaspberry Pi، Bi-LSTM، YOLOv8.