République Algérienne Démocratique et Populaire
Ministère de L'Enseignement Supérieur et de la Recherche Scientifique Université Saad Dahleb - Blida 1 -
Faculté des sciences

MÉMOIRE

Présenté pour l'obtention du **diplôme** de **Master**

En : Informatique

Option : Systèmes Informatiques et Réseaux

Thème :

Développement d'un système basé sur le Deep Learning Pour la détection des maladies végétales

Réalisé par
Serir Abdallah Omar

Encadré par
Daoud Hayat

Année universitaire : 2024/2025

Résumé

La protection des cultures agricoles représente un défi majeur, où les maladies des plantes peuvent causer des pertes de rendement significatives. Les méthodes de diagnostic traditionnelles, basées sur l'observation visuelle, sont souvent lentes, subjectives et manquent de précision, ce qui retarde les interventions critiques. Ce mémoire présente la conception et l'implémentation d'un système basé sur l'apprentissage profond pour la classification automatique de ces maladies. La solution s'appuie sur un réseau de neurones convolutif (CNN) entraîné à partir de zéro, capable d'identifier 38 classes distinctes à partir d'images de feuilles. À l'issue de la phase d'entraînement, le modèle développé a atteint une précision de 98%. Ce travail illustre ainsi le potentiel des réseaux de neurones convolutifs comme outil pour l'aide au diagnostic automatisé dans le domaine agricole.

Mots clés : Détection maladies végétales, Apprentissage Profond, Réseaux de Neurones Convolutifs, Classification d'images, Vision par ordinateur, Intelligence Artificielle, Aide à la décision.

Abstract

Protecting agricultural crops is a major challenge, as plant diseases can cause significant yield losses. Traditional diagnostic methods, based on visual observation, are often slow, subjective, and lack precision, delaying critical interventions. This thesis presents the design and implementation of a system based on deep learning for the automatic classification of these diseases. The solution is based on a Convolutional Neural Network (CNN) trained from scratch, capable of identifying 38 distinct classes from leaf images. Upon completion of the training phase, the developed model achieved an accuracy of 98%. This work thus illustrates the potential of Convolutional Neural Networks as a tool for automated diagnostic support in the agricultural domain.

Keywords: Plant Disease Detection, Deep Learning, Convolutional Neural Networks, Image Classification, Computer Vision, Artificial Intelligence, Decision Support.

ملخص

تمثل حماية المحاصيل الزراعية تحديًا رئيسيًا، حيث يمكن أن تؤدي أمراض النباتات إلى انخفاض كبير في المحصول. إن طرق التشخيص التقليدية القائمة على الفحص البصري غالبًا ما تتسم بالبطء والذاتية ونقص الدقة، مما يؤخر التدخلات الحاسمة. تعرض هذه الدراسة تصميم وتطبيق نظام يرتكز على التعلم العميق بهدف التصنيف الآلي لهذه الأمراض. الحل المقترح يعتمد على شبكة عصبونية التفافية تم تدريبها من الصفر، وهي قادرة على تمييز 38 فئة مختلفة باستخدام صور لأوراق النبات. عقب اكتمال مرحلة التدريب، بلغ النموذج المطور دقة 98%. يُبرز هذا العمل القدرات الكامنة للشبكات العصبونية الالتفافية كأداة فعالة للمساعدة في التشخيص الآلي في القطاع الزراعي

الكلمات المفتاحية: كشف أمراض النباتات، التعلم العميق، الشبكات العصبونية الالتفافية، تصنيف الصور، الرؤية الحاسوبية، الذكاء الاصطناعي، دعم اتخاذ القرار

Remerciements

Avant toute chose, nous tenons à exprimer notre gratitude envers **Allah, notre dieu**, qui nous a donné la persévérance et l'énergie nécessaires à l'accomplissement de ce mémoire.

Nous adressons nos plus vifs remerciements à notre promotrice, **Dr. Daoud Hayat**. Sa disponibilité, ses conseils avisés et son soutien constant ont été un guide précieux tout au long de ce projet.

Nos sincères remerciements vont également aux honorables membres du **jury** qui nous font l'honneur d'évaluer ce travail. Leur expertise est une précieuse contribution

Nous n'oublions pas l'ensemble de nos **professeurs** et **enseignants** qui, durant tout notre parcours universitaire, ont su partager leur savoir et ont participé à forger les connaissances qui sont à la base de ce mémoire.

Enfin, il nous est impossible de conclure sans exprimer notre plus profonde reconnaissance à nos **parents**. Leur soutien indéfectible, leur amour et leurs sacrifices ont été notre principale source de force et de motivation. Leur confiance en nous fut un pilier essentiel de notre succès, et pour cela, nous les remercions du fond du cœur.

.

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations

Abréviation	définition
IA	Intelligence Artificielle
ML	Machine Learning
ANN	Artificial Neural Networks
PE	Processing Elements
RNN	Recurrent Neural Networks
CNN	Convolutional Neural Network
GPU	Graphics Processing Unit
FC	Fully Connected
TP	True Positives
TN	True Negatives
FP	False Positives
FN	False Negatives

Introduction Générale

Contexte du projet

Depuis notre venue sur terre, les interactions avec notre environnement, notamment végétal, sont fondamentales pour notre subsistance. L'agriculture, pilier de nos sociétés, est constamment soumise à des menaces, parmi lesquelles les maladies des plantes représentent un danger majeur. Pendant des millénaires, l'Homme n'a eu de cesse de chercher des solutions et des remèdes pour protéger ses cultures. Ainsi, au fil du temps, plusieurs techniques et outils ont été développés pour aider à diagnostiquer et traiter ces affections. Il a fallu attendre l'avènement des technologies de l'information et de l'intelligence artificielle pour que chercheurs et agronomes travaillent d'arrache-pied à la mise en œuvre de systèmes aptes à détecter, à prédire et à aider au diagnostic des maladies végétales les plus destructrices.

Aujourd'hui, nous vivons une ère nouvelle, façonnée par l'intelligence artificielle, qui offre des outils de traitement et d'analyse sans précédent. Malgré toutes les innovations et les avancées technologiques, la protection des cultures contre les maladies reste un défi majeur. Si de nombreux efforts ont été faits pour enrayer ces fléaux, la rapidité et la précision du diagnostic restent des enjeux cruciaux. Un groupe de chercheurs, d'agronomes et de développeurs ont commencé à mettre au point des techniques informatiques, visant à diagnostiquer les maladies des plantes à leur tout début et ainsi, permettre au cultivateur d'avoir une chance de traiter avant que la maladie ne soit à un stade trop avancé.

L'objectif de ce projet est le développement d'un système basé sur les réseaux de neurones convolutifs, afin de détecter les maladies des plantes à partir d'images, pour l'aide à la décision.

Problématique

Les méthodes traditionnelles de diagnostic des maladies des plantes reposent souvent sur l'observation visuelle, une approche longue, subjective et parfois imprécise. Ces limites retardent les interventions, aggravant ainsi les pertes agricoles et compromettant la sécurité alimentaire, En effet le diagnostic repose souvent sur l'expérience et l'expertise des agriculteurs, Du coup les résultats varient énormément surtout là où les experts sont rares.

De plus, certaines maladies présentent des symptômes similaires, rendant la distinction difficile à l'œil nu, même pour les experts. Ces erreurs peuvent mener à des traitements inadaptés, augmentant les coûts de production et réduisant la productivité.

Il nous faut des solutions automatisées, rapides et fiables ! L'IA, notamment la vision par ordinateur et l'apprentissage profond, ouvre des perspectives géniales pour détecter les maladies des plantes directement par leurs images.

Objectifs du travail

L'objectif principal de ce mémoire est de concevoir et de développer un système intelligent d'aide au diagnostic des maladies des plantes, basé sur des techniques d'intelligence artificielle, et plus précisément sur l'apprentissage profond (deep learning). Ce système aura pour but d'analyser des images de feuilles de plantes afin d'y détecter automatiquement la présence de maladies, tout en identifiant leur type avec une précision élevée.En plus de la détection, le système proposera également des solutions adaptées ou des recommandations sur les traitements ou les actions à entreprendre, afin de guider l'utilisateur vers une prise de décision rapide et efficace.

Plan du mémoire

Pour structurer notre travail de manière claire et progressive, ce mémoire a été divisé en trois chapitres principaux :

- **Chapitre 1 : État de l'art**

  Ce chapitre établit les bases théoriques, en présentant le contexte des maladies végétales et les fondements de l'intelligence artificielle. Il explore les approches du Machine Learning et du Deep Learning, en se focalisant sur les réseaux de neurones convolutifs (CNN).

- **Chapitre 2 : Conception et spécification de la solution**

  Ce chapitre est consacré à la conception et à la spécification de notre solution. Il y est décrit l'architecture de notre modèle CNN, la préparation des données et la justification des choix techniques retenus.

- **Chapitre 3 : Implémentation et résultats**

  Ce chapitre décrit l'implémentation pratique de notre solution. Il couvre la configuration de l'environnement, le processus d'entraînement du modèle, ainsi que l'évaluation quantitative de ses performances.

I

L’état de l’art

1.1. Introduction

L’intelligence artificielle révolutionne divers secteurs en reproduisant des capacités humaines telles que l’analyse d’images. L’apprentissage profond, technique phare, excelle dans la compréhension de données visuelles complexes. En agriculture, il permet aujourd’hui une détection automatique et précise des maladies des plantes à partir de simples images. Cette avancée offre un soutien précieux aux agriculteurs pour préserver leurs cultures.

Ce chapitre explore les fondements de cette IA dédiée à la détection des maladies végétales.

1.2. Maladies Végétales

1.2.1. Définition

Une maladie végétale désigne toute altération nuisible affectant le développement, les fonctions ou l'apparence d'une plante. Ces pathologies peuvent être provoquées par divers agents pathogènes infectieux tels que champignons, bactéries, virus ou nématodes, mais aussi par des facteurs abiotiques comme des carences nutritionnelles ou des conditions climatiques extrêmes [1].

1.2.2. Causes des maladies

Les maladies végétales résultent d'une interaction nécessaire entre trois facteurs : une plante sensible, un agent pathogène spécifique, et des conditions environnementales propices au développement de la maladie. Les principaux agents pathogènes sont les champignons, les bactéries et les virus. Il est aussi reconnu que certains nématodes peuvent initier ces maladies [1].

En complément, cette interdépendance est schématisée par le "triangle de la maladie", un concept fondamental illustrant la nécessité de la présence des trois facteurs pour le développement d'une affection [2].

Ce concept fondamental modélise l'interaction entre les trois conditions nécessaires. La maladie, au centre, ne se manifeste que lorsque les trois sommets du triangle une plante hôte sensible, un agent pathogène virulent et un environnement favorable sont réunis.

Ainsi, toute stratégie de gestion des cultures vise à rompre ce triangle en agissant sur au moins un de ces trois facteurs.

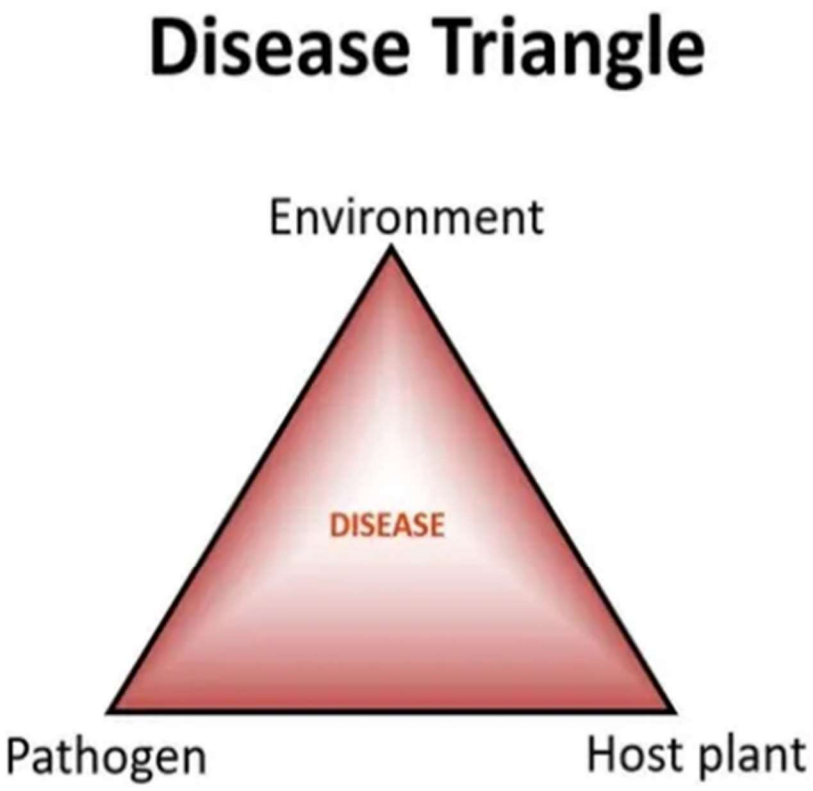

Figure 1.1 : Illustration du triangle de la maladie [2]

1.2.3. Symptômes

Le tableau suivant présente quelques symptômes fréquents observés chez les plantes atteintes de maladies.

Symptôme	**Description**	**Illustration**
Taches foliaires (Leaf spots)	Les taches foliaires apparaissent comme des marques sombres sur les feuilles, causées par des champignons, bactéries ou virus. Elles se développent surtout par temps humide prolongé [3].	

Virus de la mosaïque (Mosaic Virus)	Le virus de la mosaïque provoque des taches irrégulières vertes ou jaunes sur les feuilles, souvent accompagnées de déformations. Les plantes atteintes doivent être retirées pour limiter la propagation [3].	
Mildiou poudreux (Powdery mildew)	L'oïdium se reconnaît à une fine couche blanche semblable à de la poudre sur les feuilles et les tiges. Il se développe par temps sec et humide, surtout si l'aération entre les plantes est insuffisante [3].	
La rouille (Rust)	La rouille provoque des taches, dorées ou rougeâtres sur les feuilles. Elle affecte certaines plantes selon l'espèce, mais reste généralement peu grave si les plantes sont bien entretenues [3].	

Tableau 1.1 : Symptômes des maladies des plantes et leurs caractéristiques

1.3. Image Numérique

1.3.1. Définition

Une image numérique est une représentation visuelle dont la surface est segmentée en unités de dimensions fixes, connues sous le nom de cellules ou de pixel. Chacune de ces unités possède une valeur spécifique, correspondant soit à un niveau de gris, soit à une couleur. Cette valeur est obtenue à partir de la position correspondante dans une image réelle ou déterminée à partir d'une modélisation interne de la scène à reproduire [4].

1.3.2. Caractéristiques

Les informations d'une image sont organisées de manière structurée et se définissent par les caractéristiques suivantes :

A. Pixel

Le pixel désigne le plus petit composant d'une image. Il s'agit d'une donnée numérique représentant l'intensité lumineuse à un point précis [5].

Un pixel se compose de trois sous-pixels (rouge, vert, bleu) dont l'intensité lumineuse varie de 0 à 255 pour former une vaste gamme de couleurs. Chaque couleur d'un pixel est ainsi définie par un triplet de valeurs RVB [6].

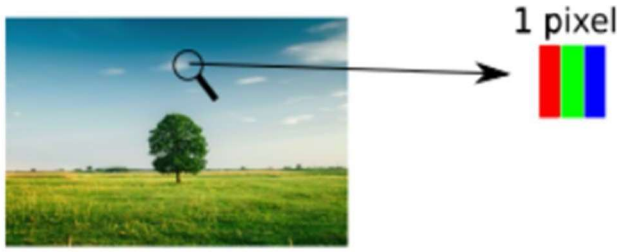

Figure 1.2 : Représentation numérique d'un pixel en couleurs RVB [6]

B. La résolution

La résolution caractérise la netteté des images produites par un écran ou une imprimante. Pour les moniteurs, elle se mesure en pixels par unité de longueur (pouce ou centimètre), ou par le nombre total de pixels affichables horizontalement et verticalement : plus ce nombre est élevé, meilleure est la résolution [4].

C. Bruit

Le bruit dans une image se manifeste par une variation soudaine de l'intensité d'un pixel par rapport à ceux qui l'entourent. Ce phénomène est souvent causé par l'éclairage des composants optiques et électroniques du capteur [4].

D. Histogramme

L'histogramme d'une image représente la fréquence de chaque niveau de gris ou couleur. Il renseigne sur la distribution tonale, notamment pour des images trop claires ou trop sombres [4]. L'histogramme est modifié pour améliorer la qualité de l'image (rehaussement), réduire les erreurs de quantification, comparer des images sous divers éclairages, ou extraire des informations.

E. Luminance

La luminance représente le degré de luminosité des points d'une image. Elle est définie comme le rapport entre l'intensité lumineuse d'une surface et son aire apparente, vue de loin [4], une bonne

luminance implique des images lumineuses, un contraste équilibré (évitant les extrêmes qui causent une perte de détails dans les zones sombres ou très claires), et l'absence de parasites (bruit visuel).

F. Contraste

Le Contraste Le contraste désigne la différence notable entre des zones claires et sombres d'une image. Il se définit par la relation entre les luminances de deux régions. Si L 1 et L 2 représentent les degrés de luminosité de deux zones adjacentes (A 1 et A 2) d'une image, le contraste C'est alors déterminé par leur rapport :

$$C = \frac{L1 - L2}{L1 + L2} \quad [4]$$

1.3.3. Les différents types d'image

On distingue trois grandes catégories d'images numériques, chacune se différenciant par sa palette de couleurs : [5]

- Images binaires : Les images binaires, également appelées noir et blanc, sont les plus élémentaires. Elles sont bichromes, ce qui signifie que leurs pixels n'ont que deux valeurs possibles, 0 ou 1. Généralement, 0 représente un pixel noir et 1 un pixel blanc, chaque niveau étant codé sur un seul bit.
- Images à niveaux de gris : Ces images représentent l'intensité lumineuse par pixel, variant du noir au blanc via de nombreuses nuances. Chaque pixel est codé sur plusieurs bits (souvent 8, soit un octet), permettant jusqu'à 256 niveaux. Plus ce nombre de bits est grand, plus la gradation est fine, ce qui exige un matériel compatible.
- Images couleurs : ces images combinent le rouge, le vert et le bleu (RVB). Chaque couleur est codée de 0 à 255 (0,0,0 pour noir ; 255,255,255 pour blanc). Elles sont représentées soit par un pixel combinant ces valeurs, soit par trois images séparées.

1.4. Intelligence Artificielle : Exploration du Machine Learning et du Deep Learning

1.4.1. Définition de L'intelligence artificiel

L'intelligence artificielle (IA) est une branche de l'informatique qui vise à doter les machines de capacités cognitives et décisionnelles semblables à celles de l'être humain. Grâce à l'IA, les machines peuvent apprendre, résoudre des problèmes complexes, prendre des décisions éclairées et même générer du contenu, simulant ainsi l'intelligence humaine pour accomplir des tâches qui lui étaient auparavant réservées [7].

L'intelligence artificielle transforme le secteur agricole en rendant possible une gestion de très haute précision. Grâce à l'analyse de données en temps réel les agriculteurs peuvent déterminer avec exactitude quelles parcelles de leur exploitation requièrent une intervention ciblée, que ce soit pour l'apport en eau, en fertilisants ou en traitements phytosanitaires, cette gestion optimisée des ressources aboutit à des bénéfices concrets : l'usage des herbicides est rationalisé, la qualité des récoltes est améliorée et la rentabilité globale de l'exploitation est accrue par une augmentation des profits et une diminution des coûts[29].

1.4.2. Histoire de l'IA : Jalons Clés

Le tableau ci-dessous, présente une chronologie des principaux événements et découvertes qui ont façonné l'évolution de l'Intelligence Artificielle [8].

Année	Événement	Personnalités clés	Description
1956	Le Dartmouth Workshop	John McCarthy, Marvin Minsky, Nathaniel Rochester, Claude Shannon	Généralement perçu comme l'événement fondateur de l'Intelligence Artificielle en tant que discipline scientifique distincte, cet atelier de deux mois a établi les fondements de la recherche en IA explorant des domaines essentiels tels que la résolution de problèmes, l'apprentissage automatique et le traitement du langage et ouvrant la voie à des avancées futures majeures dans le domaine.
1957	Le Perceptron	Frank Rosenblatt	Figure marquante des précurseurs des réseaux neuronaux, il a marqué un tournant en introduisant les perceptrons, des réseaux à une seule couche, aptes à l'apprentissage et à la décision. Bien que rudimentaires, ils ont jeté les bases essentielles des recherches ultérieures en réseaux neuronaux.
1966	Eliza	Joseph Weizenbaum	Conçu comme l'un des tout premiers agents conversationnels, ELIZA avait pour mission de simuler des échanges humains. Bien que ses interactions puissent paraître simplistes au regard des standards actuels, ce programme a démontré de manière significative le potentiel du traitement du langage naturel (TLN) en IA.

1970s-1980s	L'ère des Systèmes Experts	Multiples chercheurs et équipes (ex : Université de Stanford pour MYCIN)	Durant ces décennies, les systèmes experts se sont imposés comme une branche prédominante de l'IA. Ces applications informatiques étaient conçues pour mimer le raisonnement d'un spécialiste humain dans des domaines spécifiques, en utilisant des bases de règles.MYCIN est un exemple emblématique, ayant prouvé la capacité de ces systèmes à diagnostiquer des maladies infectieuses avec une grande fiabilité.
1997	Deep Blue contre Garry Kasparov	Deep Blue (superordinateur IBM), Garry Kasparov	L'année 1997 fut le théâtre d'un affrontement emblématique qui redéfinit la perception de l'IA. La victoire de Deep Blue, le superordinateur d'IBM, contre Garry Kasparov (champion du monde d'échecs) a démontré de façon spectaculaire qu'une machine pouvait non seulement égaler, mais dépasser l'intelligence humaine dans un domaine aussi complexe que les échecs.
1997	Formalisation de l'Apprentissage Automatique	Tom Mitchell	En 1997, Tom Mitchell, éminent informaticien, a proposé une définition fondatrice de l'apprentissage automatique : un champ scientifique consacré à la conception et l'analyse d'algorithmes apprenant à partir de données. Cette conceptualisation a marqué un tournant dans la recherche en IA, mettant en lumière le rôle crucial des algorithmes alimentés par les données et ouvrant la voie à des systèmes capables d'auto-adaptation et d'optimisation progressive.
2012	ImageNet et l'Avènement du Deep Learning	Équipe de Geoffrey Hinton	Le challenge ImageNet a marqué un tournant en reconnaissance visuelle. L'équipe de Geoffrey Hinton a révolutionné la précision des algorithmes grâce aux réseaux convolutifs (CNN). Ce succès a relancé l'intérêt pour l'IA, montrant que le deep learning pouvait exploiter de grandes quantités de données pour des résultats impressionnants en classification d'images.

2014	Réseaux Génératifs Adversariaux (GANs)	Ian Goodfellow	Les GANs, introduits par Ian Goodfellow en 2014, ont révolutionné le domaine de la modélisation générative Ce cadre novateur repose sur la compétition de deux réseaux neuronaux – un générateur, qui crée des données, et un discriminateur, qui évalue leur réalisme, produisant ainsi des résultats d'un réalisme inédit en synthèse d'images et de contenus multimédias."
2017	L'Émergence des Transformers en TLN	Vaswani et al.	L'année 2017 a été marquée par l'introduction du modèle Transformer, une architecture révolutionnaire, qui a profondément transformé le domaine du traitement du langage naturel (TLN). Les Transformers se distinguent par l'utilisation de mécanismes d'auto-attention, leur permettant de traiter des séquences de mots avec une efficacité inédite pour la compréhension et la génération du langage.

Tableau 1.2 Événements Marquants de l'Histoire de l'IA [8]

1.5. Machine learning

1.5.1. Définition

L'apprentissage automatique (ML) est une branche de l'intelligence artificielle (IA) qui vise à donner aux ordinateurs la capacité d'apprendre de manière similaire à l'être humain. Son objectif est de leur permettre d'exécuter des tâches de façon autonome et de perfectionner leur efficacité et leur justesse à mesure qu'ils acquièrent de l'expérience et traitent davantage de données [9].

Un algorithme d'apprentissage automatique supervisé repose généralement sur trois éléments fondamentaux [10] :

- Le Processus de Décision :

Cette phase regroupe les calculs et les procédures par lesquels l'algorithme examine les informations fournies pour inférer le modèle ou la structure sous-jacente qu'il doit identifier.

- La Fonction d'Erreur (ou de Coût) :

Cette composante sert à évaluer la justesse de la prédiction du modèle. Elle y parvient en comparant la "devinette" de l'algorithme aux exemples réels connus (lorsqu'ils existent), permettant ainsi de mesurer précisément l'ampleur de l'écart ou de l'inexactitude.

- Le Processus d'optimisation :

Cette étape cruciale intervient lorsque l'algorithme a identifié une erreur. Il analyse alors cet écart et ajuste la manière dont le processus de décision aboutit à sa prédiction, dans le but de minimiser cette erreur lors des tentatives futures.

1.5.2. Types du machine learning

Le domaine de l'apprentissage automatique englobe différentes approches. Celles-ci seront présentées ici :

- **Apprentissage supervisé :**

L'apprentissage supervisé consiste à entraîner un algorithme à partir de données étiquetées, nécessitant une intervention humaine importante. Les données sont divisées en caractéristiques (entrées) et en étiquettes (sorties), ces dernières étant souvent annotées manuellement. Ce type d'apprentissage s'améliore au fil du temps grâce aux retours humains et à l'ajout de nouvelles données [11].

On distingue deux principaux types d'algorithmes en apprentissage supervisé :

- Classification : Les algorithmes de classification sont utilisés pour analyser et catégoriser automatiquement des données, comme dans le tri des e-mails (KNN, SVM)
- Regression : Les algorithmes de régression servent à prédire des tendances en modélisant les relations entre variables (Régression linéaire, Régression logistique).

- **Apprentissage non supervisé :**

L'apprentissage non supervisé utilise des algorithmes d'IA pour détecter des motifs dans des données non étiquetées, sans intervention humaine. Il permet de regrouper, classer ou segmenter automatiquement les données selon leurs similarités [12].

- **Apprentissage semi-supervisé :**

L'apprentissage semi-supervisé combine une petite quantité de données étiquetées avec un grand volume de données non étiquetées pour entraîner un modèle. Il est utile lorsque l'étiquetage est coûteux ou long, mais que les données non étiquetées sont abondantes [13].

- **Apprentissage par renforcement :**

L'apprentissage par renforcement permet à un agent d'apprendre à prendre des décisions en interagissant avec un environnement. Il reçoit des récompenses ou des pénalités en fonction de ses actions, ce qui l'aide à améliorer progressivement son comportement [14].

La figure présente les quatre types d'apprentissage automatique et quelques exemples d'utilisation.

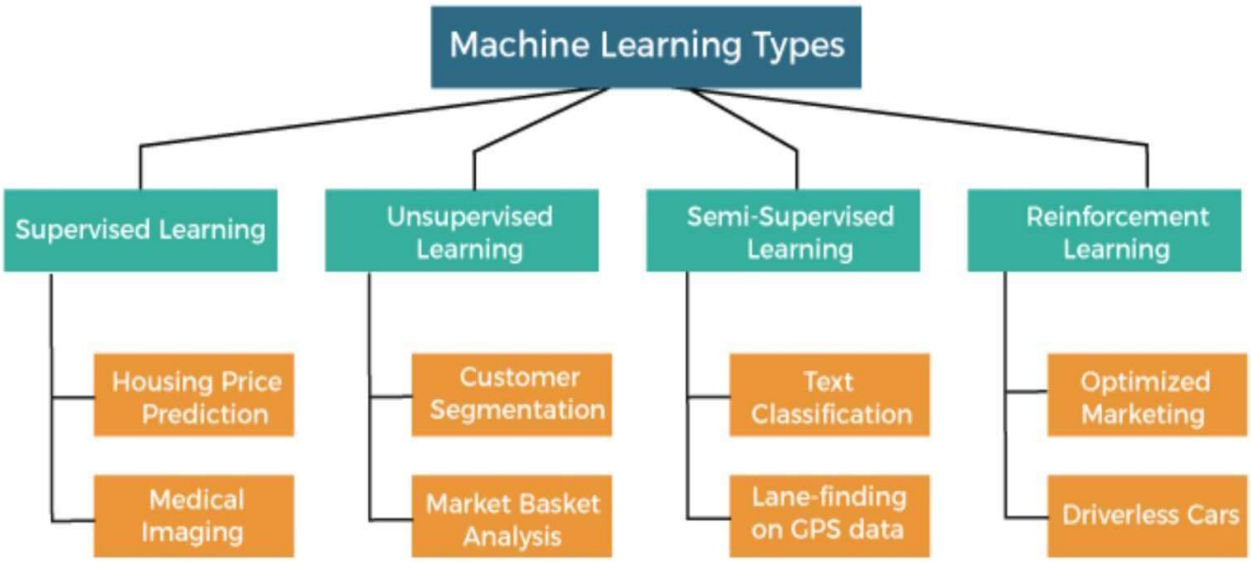

Figure 1.3 : Types d'apprentissage automatique et quelques exemples d'utilisation [15]

1.6. Réseaux de Neurones Artificiels

1.6.1. Définition

Les réseaux de neurones artificiels (ANNs) sont des systèmes adaptatifs inspirés par le fonctionnement du cerveau humain. Ils sont capables de modifier leur structure interne pour atteindre un objectif fonctionnel. Ces systèmes sont particulièrement adaptés pour résoudre des problèmes non linéaires, car ils peuvent reconstituer les règles complexes qui régissent la solution optimale [16].

Les éléments fondamentaux d'un ANN sont les nœuds, également appelés éléments de traitement (PE), et les connexions entre eux. Chaque nœud possède sa propre entrée, par laquelle il reçoit des informations d'autres nœuds ou de l'environnement, et sa propre sortie, par laquelle il communique avec d'autres nœuds ou avec l'environnement. De plus, chaque nœud est doté d'une fonction f qui lui permet de transformer son entrée globale en une sortie.

La figure suivante illustre un schéma d'un unique élément de traitement (PE) :

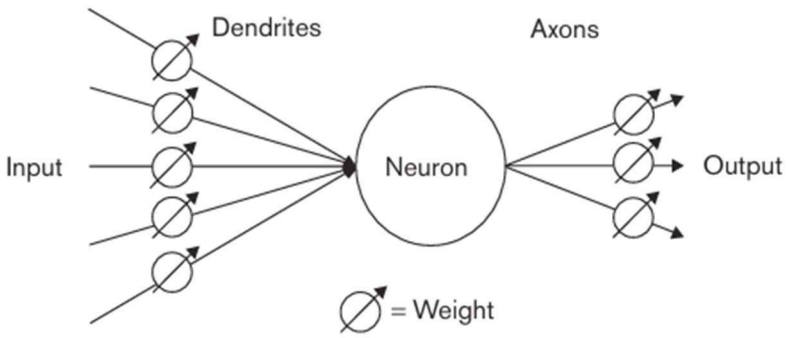

Figure 1.4 : Diagramme d'un unique élément de traitement (PE) [16]

Chaque connexion d'un ANN possède un poids qui détermine l'excitation ou l'inhibition entre les nœuds. Ces poids se modifient via la "Loi d'Apprentissage", permettant au réseau d'adapter ses connexions pour comprendre la structure des données.

1.6.2. Topologie des ANN : Le modèle à propagation avant

Les réseaux neuronaux artificiels (ANNs) peuvent être organisés selon diverses configurations topologiques, leur arrangement dépendant de la nature et du volume des données en entrée. La topologie la plus fréquemment rencontrée chez les ANNs est celle dite "à propagation avant" (feed-forward). Dans ce modèle, une couche d'entrée est constituée d'éléments de traitement (PEs), leur nombre étant généralement défini par la quantité de variables d'entrée. L'information transite ensuite vers une ou plusieurs couches cachées opérant au sein du réseau. Enfin, une couche de sortie fournit le résultat final [16].

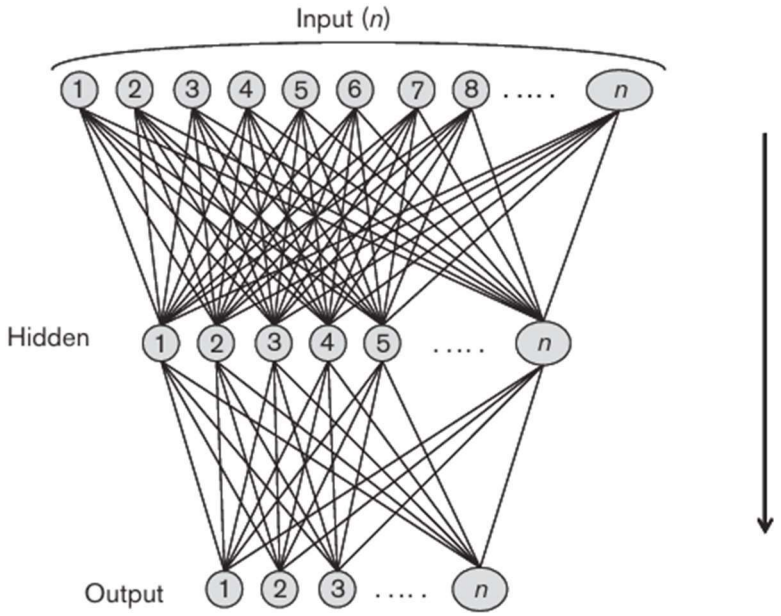

Figure 1.5 : Architecture d'un réseau de neurones à propagation avant [16].

La Figure 1.5 présente cette architecture, les éléments de traitement (PEs) sont interconnectés, chaque connexion ayant un "poids" qui est ajusté pendant l'entraînement. Ce processus permet au réseau d'apprendre à classer les données sans règles pré-établies.

1.7. Introduction au deep learning

Le deep learning, ou apprentissage profond, constitue un domaine du machine learning qui exploite des réseaux neuronaux profonds composés de plusieurs couches. Ces structures permettent de reproduire la capacité du cerveau humain à effectuer des raisonnements complexes. De nombreuses applications d'intelligence artificielle (IA) actuelles reposent sur cette technologie [17].

1.7.1. Différences entre Machine learning et Deep learning en Agriculture

L'agriculture moderne fait de plus en plus appel à l'intelligence artificielle pour optimiser les rendements, lutter contre les maladies. Deux approches se distinguent particulièrement : l'apprentissage Automatique (Machine Learning) traditionnel et l'apprentissage Profond (Deep Learning). Bien que le second soit une sous-catégorie du premier, leurs applications, exigences et méthodologies en agriculture présentent des différences notables, le tableau suivant met en évidence ces distinctions clés particulièrement pertinentes dans le cadre de la détection des maladies végétales.

Critère	Machine Learning	Deep Learning
Besoin d'intervention humaine	Requiert une étape de "sélection de caractéristiques manuelle. Un expert doit définir les traits pertinents (ex : texture, couleur, forme) à extraire des données [26].	Apprend et extrait automatiquement les caractéristiques pertinentes directement à partir des données brutes (ex : pixels d'une image), réduisant le besoin d'expertise humaine pour cette étape [27].
Temps d'entraînement	Court : quelques minutes à quelques heures [18]	Long et coûteux en calcul [28].
Matériel requis	Peut s'exécuter sur des machines de faible à moyenne puissance [31].	Exige quasi systématiquement des processeurs graphiques (GPU) puissants pour gérer la complexité des calculs des réseaux de neurones profonds [28].
Cas d'usage fréquents	Prédiction des rendements agricoles, gestion de l'irrigation basée sur des données de capteurs, classification des types de sols [26].	Détection et classification des maladies des plantes, identification des mauvaises herbes, comptage de fruits, agriculture de précision par drones et analyse d'images satellites [27].

Tableau 1.3 Tableau comparatif entre Machine Learning et Deep Learning en Agriculture

1.7.2. Modèles des réseaux de neurones en apprentissage profond

❖ Les réseaux de neurones récurrents

Les réseaux de neurones récurrents (RNN) sont des modèles d'intelligence artificielle conçus pour traiter des données séquentielles. Ils se distinguent par leur capacité à réutiliser leurs sorties passées comme entrées futures, ce qui leur permet de « mémoriser » le contexte. Cette architecture est particulièrement adaptée aux tâches impliquant des séries temporelles ou du texte. Grâce à leurs boucles internes, les RNN ajustent leurs prédictions en tenant compte des éléments précédents. Toutefois, cette flexibilité rend leur entraînement plus complexe [19].

❖ Réseaux de neurones convolutifs

Les réseaux neuronaux convolutifs (ou ConvNets) sont une catégorie d'algorithmes d'apprentissage profond spécifiquement conçus pour exceller dans les tâches de reconnaissance d'objets. Leur architecture unique les rend particulièrement efficaces pour des applications comme la classification, la détection et la segmentation d'images. On les retrouve notamment dans les véhicules autonomes, dans les systèmes de vidéosurveillance et dans bien d'autres domaines où l'analyse visuelle est cruciale. [20].

L'architecture d'un réseau de neurones convolutifs repose sur une succession de blocs de traitement, dont l'objectif est d'extraire les caractéristiques pertinentes permettant de différencier la classe de l'image des autres. Chaque bloc est constitué d'une ou plusieurs couches chargées de transformer les données en représentations de plus en plus discriminantes.

Pour mieux comprendre son fonctionnement, la Figure 1.6 ci-dessous illustre l'architecture typique d'un CNN et le cheminement des données à travers ses différentes couches.

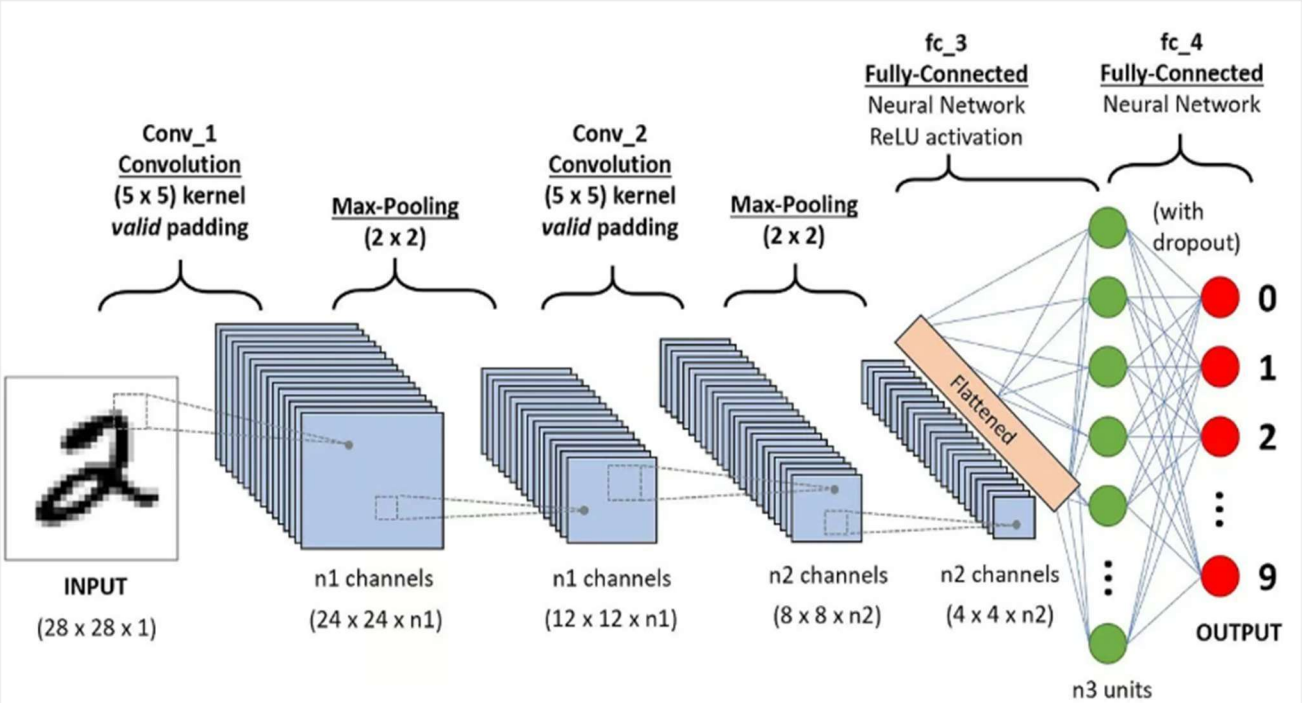

Figure 1.6 : Architecture d'un réseau de neurones convolutif [20]

Comme l'illustre la Figure 1.6, le parcours des données au sein du réseau est une transformation progressive. Le système part d'une information visuelle concrète pour aboutir à une conclusion abstraite.

Dans la première partie du réseau, on observe que l'image d'entrée traverse plusieurs blocs de traitement. À chaque étape, la représentation des données devient physiquement plus petite en hauteur et en largeur, mais gagne en "profondeur", comme l'indique l'empilement croissant des cartes de caractéristiques. Le réseau se focalise ainsi sur une information de plus en plus concentrée.

Une étape de transition cruciale intervient ensuite, où cette structure tridimensionnelle de cartes est réorganisée en une seule et longue séquence d'informations : un vecteur unidimensionnel.

Ce vecteur alimente la seconde partie du réseau, la tête décisionnelle. Ces dernières couches analysent cette séquence de données dans sa globalité pour en déduire les liens logiques. Le processus se termine par la production d'un ensemble de scores de probabilité, un pour chaque classe possible, permettant au modèle de formuler sa prédiction finale.

- ## Couche de convolution

Au sein d'un réseau neuronal convolutif (CNN), la couche de convolution est l'élément central où s'opèrent les convolutions. Cette couche reçoit une image (ou un vecteur d'entrée), y applique des filtres (aussi appelés détecteurs de caractéristiques), et génère une carte de caractéristiques (feature map). En clair :

$$\text{Image d'entrée} \times \text{Filtre} = \text{Carte de caractéristiques}$$ [22]

Son rôle est d'identifier et d'extraire les motifs et caractéristiques les plus pertinents de l'image. Pour ce faire, la représentation matricielle de l'image est multipliée élément par élément avec les filtres, et les résultats sont ensuite sommés pour former la carte de caractéristiques, un processus similaire à un produit scalaire.

La convolution se distingue par des propriétés essentielles :

- **Connectivité locale** : Chaque neurone de cette couche ne traite qu'une petite portion de l'image. Le filtre, de taille définie, glisse sur ces sous-ensembles, permettant à divers filtres d'apprendre différentes facettes de l'image.
- **Partage de paramètres** : Tous les neurones d'une même carte de caractéristiques partagent les mêmes poids. C'est le même filtre qui est appliqué sur l'ensemble de l'image, ce qui réduit considérablement le nombre de paramètres à apprendre et rend le modèle plus efficace.

Ces propriétés définissent l'essence de la convolution. Cependant, la manière dont cette opération est appliquée, notamment en termes de dimensions et de couverture de l'image, est affinée par des techniques comme le padding et le stride :

- **Padding** (Remplissage) : consiste à ajouter des zéros aux bordures de la matrice d'entrée d'une image. Comme illustré dans la Figure 1.7, Cela permet de maintenir la taille de la sortie, d'optimiser la couverture par le filtre, et de préserver les informations cruciales des bords de l'image.

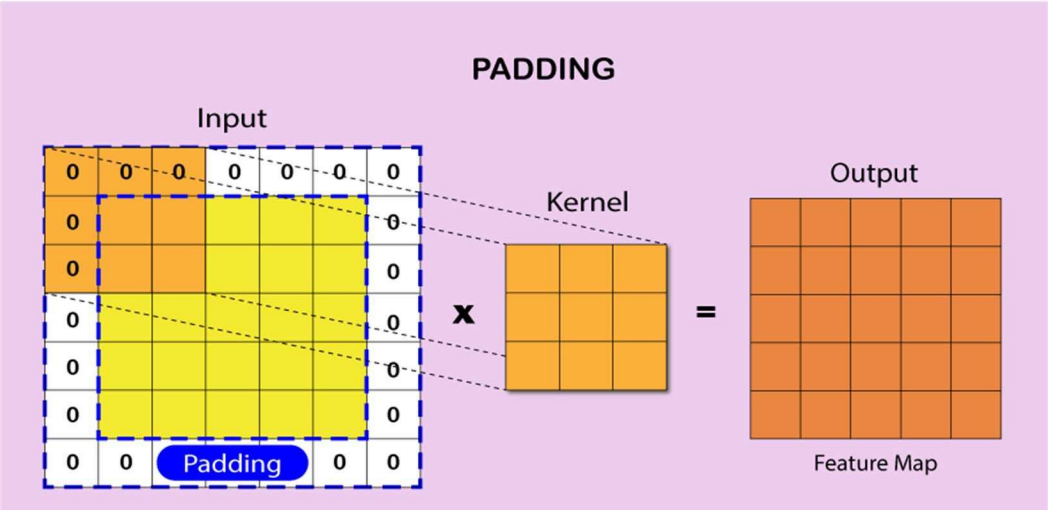

Figure 1.7 : Illustration du Padding dans une Couche de Convolution [22]

- **Stride** (Pas) : détermine le déplacement du filtre sur la matrice d'entrée, définissant le nombre de pixels par lesquels il se décale. Comme visible sur la Figure 8, un stride de 1 implique un mouvement d'un pixel à la fois, tandis qu'un stride de 2 signifie un déplacement de deux pixels. Une valeur de stride plus élevée résulte en une carte de caractéristiques de sortie plus petite.

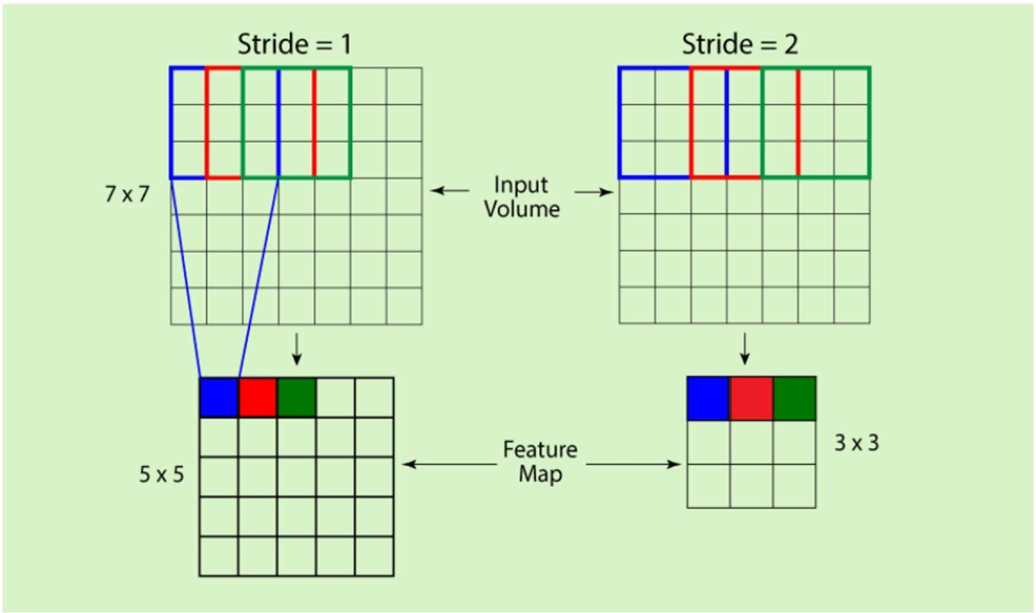

Figure 1.8 : Illustration du Stride dans une Couche de Convolution [22]

- **Couche de pooling**

Cette couche est une étape qui suit généralement la couche de convolution. Elle agit sur chaque carte de caractéristiques individuellement. Son rôle est de réduire la résolution des cartes en diminuant leur hauteur et leur largeur, tout en conservant les caractéristiques clés nécessaires à la classification. Ce processus est donc un sous-échantillonnage.

Il existe principalement deux méthodes de pooling, comme illustré par la Figure 1.9 :

- **Max-pooling** : consiste à extraire la valeur maximale de chaque région d'une carte de caractéristiques, conservant ainsi les éléments les plus significatifs. C'est la méthode la plus répandue en raison de son efficacité.
- **Average pooling** : calcule quant à lui la moyenne des valeurs dans chaque zone de la carte, offrant une représentation plus lissée des caractéristiques.

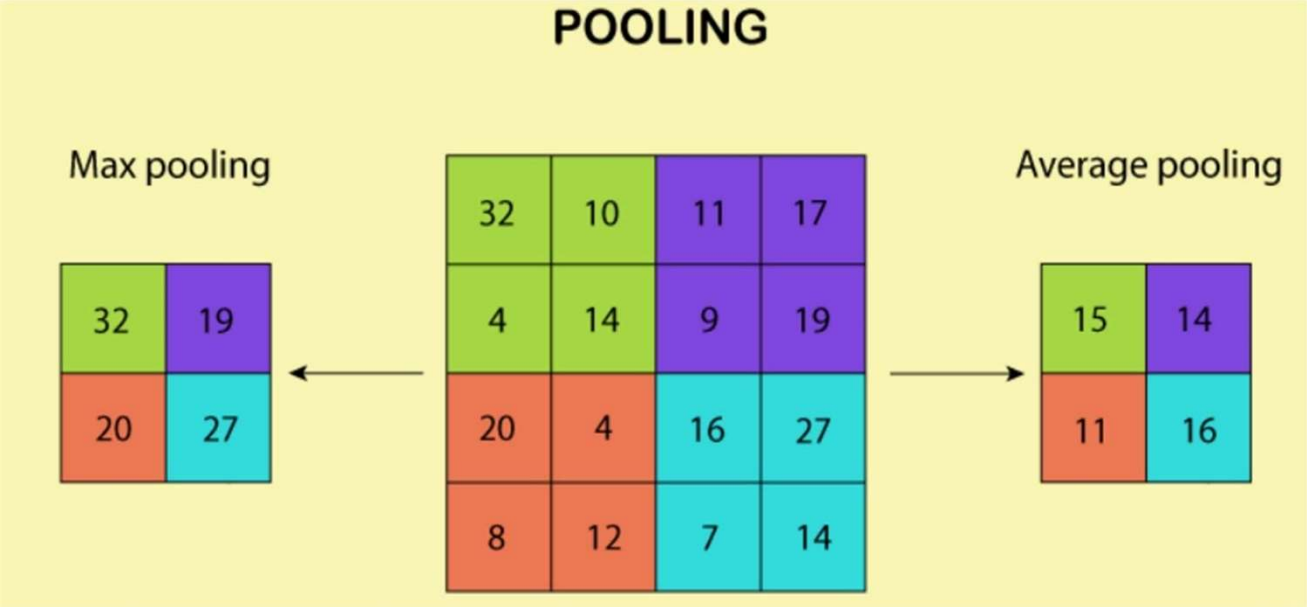

Figure 1.9 : Comparaison du Max-pooling et de l'Average pooling [22]

- **Couche de correction**

Cette couche introduit une non-linéarité essentielle en appliquant une fonction (comme ReLU) aux données issues de la couche de pooling. Cela permet au modèle de saisir des motifs plus complexes dans les données d'entrée [23].

- **Couche entièrement connectée**

La couche entièrement connectée (FC) sert de phase finale dans l'architecture d'un CNN. Caractérisée par des connexions complètes entre ses neurones et ceux de la couche précédente, elle reçoit un vecteur d'entrée. Son rôle est de classer l'image en appliquant une combinaison linéaire suivie d'une fonction d'activation. Le résultat est un vecteur indiquant la probabilité de l'image d'appartenir à chacune des classes définies [24].

- **Couche Dropout**

Cette couche est une technique de **régularisation** essentielle pour contrer le surapprentissage et améliorer la précision en test. Il fonctionne en désactivant aléatoirement (avec une probabilité p) des connexions entre les neurones d'une couche à la suivante, pour chaque mini-lot d'entraînement [25].

1.8. Utilité des CNN en agriculture

Grâce à leur architecture optimisée pour le traitement d'images, les réseaux de neurones convolutifs (CNN) se sont imposés comme des outils performants pour l'analyse de données visuelles leur capacité à apprendre automatiquement des caractéristiques complexes à partir de vastes ensembles de données leur permet de détecter et d'interpréter des motifs subtils. Ces qualités font des CNN des solutions particulièrement adaptées à un large éventail d'applications agricoles.

Les réseaux de neurones convolutifs (CNN) sont largement utilisés pour la détection des maladies des plantes une problématique majeure en agriculture. En analysant des images de végétaux, ils permettent d'identifier rapidement les symptômes de maladies, limitant ainsi leur propagation et réduisant les pertes de rendement. Cette approche facilite des interventions ciblées et efficaces dans la gestion des cultures [30].

1.9. Travaux connexes

Plusieurs travaux ont démontré l'efficacité des réseaux de neurones convolutifs (CNN) pour l'identification des maladies des plantes, chacun avec des approches et des défis spécifiques.

Parmi les études notables, les travaux de Mohanty et al. (2016) sur le jeu de données PlantVillage ont servi de référence, bien que leur applicabilité en conditions réelles ait été questionnée. D'autres chercheurs se sont concentrés sur des conditions de terrain, comme Fuentes et al. (2017) pour la tomate, mais ont fait face à des limitations dues au nombre d'échantillons.

Des recherches sur le riz et le maïs par Chen et al. (2020) ont mis en avant le défi du déploiement sur des applications mobiles pour un usage concret. Pour surmonter les limites des jeux de données, des approches innovantes ont été explorées, comme celle de Bin et al. (2017) sur les feuilles de pommier, qui ont utilisé la génération d'images pour améliorer leur modèle [29].

1.10. Conclusion

Ce premier chapitre a établi les fondements théoriques nécessaires à la compréhension de notre étude en partant de la définition des maladies végétales et de l'importance de leurs symptômes visuels, nous avons exploré les principes de l'imagerie numérique, qui constitue le support de nos données, L'analyse de l'état de l'art a ensuite mis en évidence l'évolution des techniques d'intelligence artificielle, en passant du Machine Learning traditionnel au Deep Learning, pour aboutir aux réseaux de neurones convolutifs (CNN) comme technologie de pointe pour l'analyse d'images.

L'analyse de l'état de l'art révèle que les CNN se sont imposés comme l'approche la plus performante pour l'identification automatisée des maladies des plantes. Leur capacité à apprendre hiérarchiquement des caractéristiques complexes directement à partir des pixels, sans nécessiter d'extraction manuelle, leur confère une efficacité et une précision inégalées.

Le chapitre suivant sera ainsi dédié à la présentation de notre propre système. Il s'attachera à décrire l'architecture du modèle proposé et la méthodologie mise en œuvre pour sa conception.

II

Conception et spécification de la solution

2.1. Introduction

Ce chapitre est dédié à la conception et à la spécification de l'architecture du réseau de neurones convolutifs (CNN) que nous avons développé pour aborder la problématique cruciale de la détection des maladies des plantes. Face aux défis que représentent ces maladies pour l'agriculture et la sécurité alimentaire, la mise en place de systèmes de détection précis et efficaces est primordiale. Alors que le Chapitre 1 a exploré l'état de l'art et les fondements théoriques des réseaux de neurones profonds, en particulier les CNNs, ce chapitre se concentrera sur l'application concrète de ces principes à la création d'un modèle performant et robuste.

La conception d'une architecture de réseau de neurones est une étape déterminante qui influence directement les capacités d'apprentissage, la précision et l'efficacité du modèle. Elle implique des choix stratégiques concernant le nombre et le type de couches, leur agencement séquentiel, ainsi que la définition des hyperparamètres clés. Notre modèle a été conçu pour être capable d'extraire automatiquement des caractéristiques pertinentes à partir des images foliaires des plantes, et de les interpréter afin d'identifier la présence et le type de maladie.

2.2. Architecture générale du système

La figure ci-dessous illustre l'architecture globale de notre solution, articulée autour de deux composantes majeures : le prétraitement des images et la modélisation par CNN. Cette approche permet de transformer des données brutes en diagnostics précis.

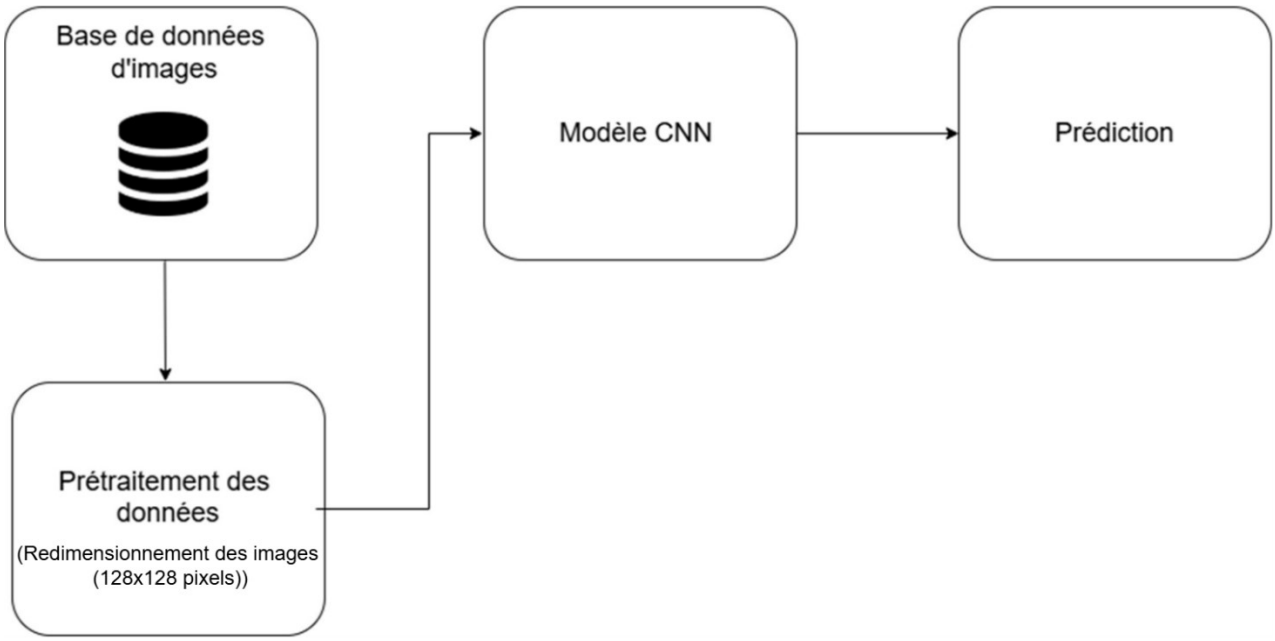

Figure 2.1 : Pipeline du Système de Détection des Maladies Végétales

Notre méthodologie se décompose comme suit :

- **Prétraitement** : Préparation des images pour l'apprentissage (redimensionnement, normalisation, augmentation).
- **Modélisation** : Architecture CNN optimisée pour la détection des maladies des plantes.

2.3. Prétraitement des données

Nous définirons dans cette section les différentes notions relatives à la préparation des images pour l'apprentissage, en nous concentrant sur le redimensionnement, la normalisation et l'augmentation des données. L'objectif est d'optimiser notre ensemble de données pour l'entraînement de notre modèle.

- **Redimensionnement**

Le jeu de données utilisé dans ce projet contenait des images ayant initialement une résolution de 256×256 pixels. Nous avons choisi de les redimensionner à 128×128 pixels afin de réduire la charge computationnelle, d'accélérer l'entraînement du modèle et de limiter l'utilisation des ressources matérielles.Ce format plus compact conserve suffisamment d'informations visuelles pour permettre une détection efficace des maladies tout en facilitant le traitement.

- **Normalisation**

Les images exploitées dans ce projet sont en couleurs (format RGB), avec des valeurs de pixels comprises entre 0 et 255. Une étape courante dans les projets d'apprentissage profond consiste à ramener ces valeurs dans une plage plus réduite, typiquement entre 0 et 1, afin de faciliter l'optimisation et d'assurer une convergence plus rapide et plus stable du modèle.

Dans notre cas, cette transformation n'a pas été appliquée lors de l'entraînement initial du modèle. Cette décision s'appuie sur une observation empirique : dès les premières phases d'entraînement, le modèle a présenté des performances satisfaisantes, avec des taux de précision élevés et une perte bien maîtrisée, malgré l'absence de normalisation explicite. Nous avons donc choisi de maintenir cette configuration simple, afin de nous concentrer sur d'autres aspects essentiels de la conception du modèle, tels que l'architecture du réseau, la qualité des données, et le surapprentissage.

- **Augmentation des données**

L'augmentation de données crée de nouveaux échantillons de données à partir de données existantes afin d'améliorer l'optimisation et la généralisation des modèles.Cette technique permet d'accroître la taille et la diversité d'un jeu de données, réduisant ainsi le surajustement et améliorant la robustesse du modèle. Elle est particulièrement utile pour pallier le manque de données réelles et pour corriger les jeux de données déséquilibrés .

Dans le cadre de ce projet, aucune technique d'augmentation de données n'a été appliquée sur le dataset. Ce choix s'explique par plusieurs considérations :

- Le jeu de données utilisé contenait déjà un volume d'images suffisant pour entraîner un modèle efficace, avec une distribution relativement équilibrée entre les différentes classes de maladies.
- Enfin, l'absence d'augmentation a permis de simplifier le pipeline de traitement et de limiter les risques d'erreurs ou de surcomplexité, notamment pour un premier prototype fonctionnel du modèle.

2.4. Conception du modèle CNN

Dans le cadre de ce projet, un modèle de réseau de neurones convolutif (CNN) a été conçu et entraîné à partir de zéro (from scratch) pour la détection automatique des maladies des plantes. Un modèle séquentiel permet de construire un réseau couche par couche, de manière linéaire, ce qui facilite l'expérimentation et l'ajustement des paramètres.

Nous avons opté pour un entraînement complet sur notre propre jeu de données. Cette stratégie a été choisie pour trois raisons principales :

- **La spécificité du domaine d'application :** Il existe un écart de domaine important entre les images génériques d'ImageNet (animaux, objets, etc.) et les images spécifiques de notre projet. Les caractéristiques visuelles des maladies des plantes (textures des lésions, motifs de décoloration) sont très particulières. Un entraînement à partir de zéro force le modèle à apprendre des représentations directement pertinentes pour notre tâche, sans être biaisé par des connaissances d'un domaine trop éloigné.La bonne performance obtenue dès les premières itérations, sans recours à des modèles pré-entraînés.

- **La flexibilité d'une architecture sur mesure :** L'entraînement (from scratch) nous a permis de concevoir une architecture CNN séquentielle parfaitement adaptée à la complexité de notre problème. Plutôt que d'utiliser une architecture pré-entraînée, souvent très lourde et complexe, nous avons pu construire un modèle couche par couche, optimisant le nombre de paramètres et facilitant l'expérimentation pour atteindre un équilibre idéal entre performance et efficacité.

- **La taille suffisante du jeu de données utilisé :** L'apprentissage par transfert est une méthode généralement utilisée lorsque le jeu de données est trop limité pour entraîner un modèle complet à partir de zéro. Le jeu de données exploité dans ce projet étant suffisamment conséquent et spécifique, cette contrainte est levée. La quantité de données disponibles est suffisante pour permettre au modèle d'apprendre les caractéristiques pertinentes directement, sans dépendre des connaissances pré-acquises d'un modèle générique.

2.4.1. Description détaillée de l’architecture CNN

Pour ce projet, nous avons conçu une architecture de réseau de neurones convolutif (CNN) basée sur un **modèle séquentiel**. Ce dernier prend en entrée des tenseurs de forme (128, 128, 3), correspondant à des images RGB de 128x128 pixels. La structure que nous avons mise en place peut être divisée en deux sous-ensembles fonctionnels : **un corps d'extraction de caractéristiques** et **une tête de classification**.

Avant de détailler chaque composant, la figure ci-dessou (2.2) offre une représentation visuelle schématique du modèle complet. Ce diagramme illustre le flux de données, depuis l'image d'entrée jusqu'à la classification finale, en passant par les blocs d'extraction de caractéristiques et la tête de classification. Les différentes couleurs permettent d'identifier facilement chaque type de couche, conformément à la légende.

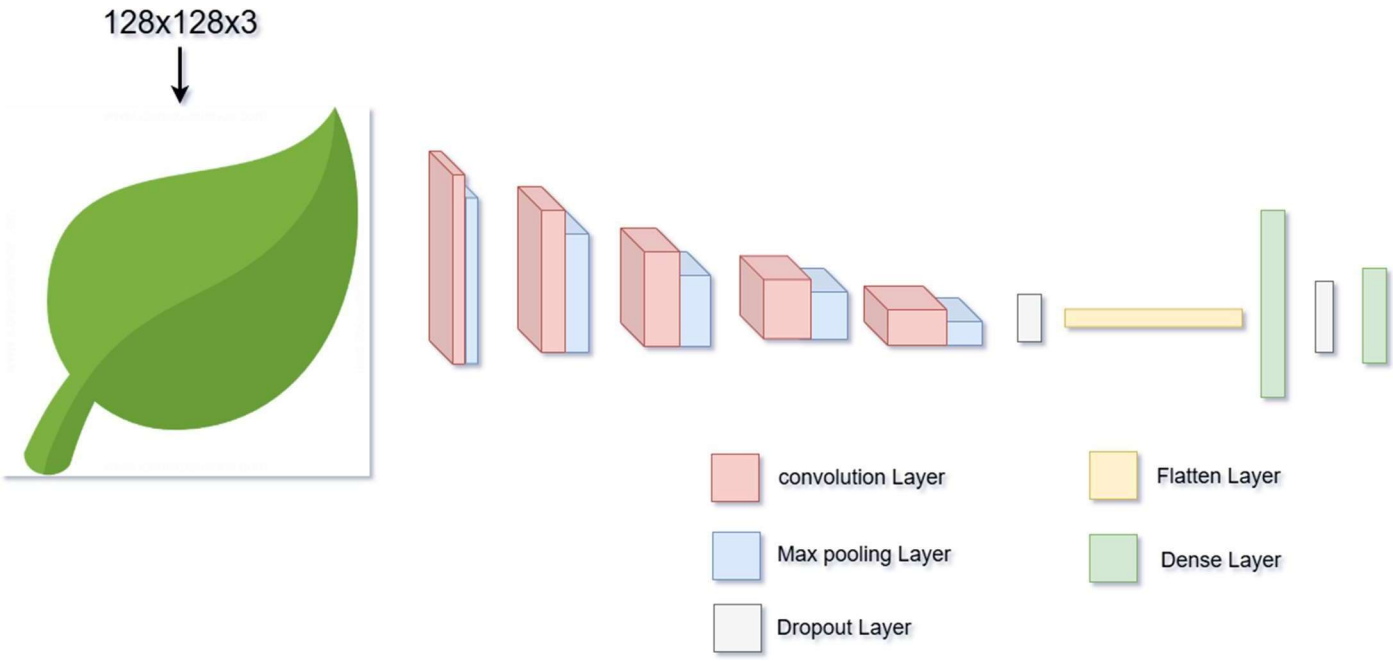

Figure 2.2 : Schéma de l'Architecture du Modèle CNN Conçu

❖ **Le Corps d'Extraction de Caractéristiques**

Cette première partie du réseau effectue un processus d'analyse en plusieurs étapes, comme illustré par les blocs roses (Convolution) et bleus (Max Pooling) dans la figure 2.2. L'objectif est de prendre une image large et riche en pixels et de la faire passer à travers cinq blocs de traitement pour en extraire l'essence sémantique dans un format compact. Le fonctionnement de chaque bloc repose sur deux actions clés, appliquées dans un ordre précis :

1. L'Enrichissement Sémantique **:** Une couche Conv est appliquée pour augmenter la profondeur de l'analyse. Elle utilise un nombre croissant de filtres pour capturer des caractéristiques de plus en plus complexes à partir de l'image.
2. La Compression Spatiale : Immédiatement après, une couche MaxPooling est utilisée pour réduire la taille (hauteur et largeur) des cartes de caractéristiques.

L'analyse se déroule en plusieurs étapes clés :

- **Étape 1** : Le Filtrage Initial (Bloc 1 avec 32 filtres)

Au début du processus, le modèle analyse l'image dans sa totalité pour détecter les caractéristiques les plus fondamentales. Les 32 filtres de ce premier bloc sont conçus pour agir comme des détecteurs de primitives : ils identifient les contours, les lignes et les changements de couleur les plus simples. C'est la première passe de filtrage, qui extrait les informations visuelles brutes.

- **Étape 2** : Les Étapes Intermédiaires (Blocs 2 et 3 avec 64 et 128 filtres)

À mesure que l'information progresse dans le réseau, sa dimension spatiale se réduit. Pour compenser, le modèle approfondit son analyse. Les blocs 2 et 3, avec un nombre de filtres plus élevé, apprennent à combiner les primitives de l'étape précédente pour former des motifs plus significatifs, comme des textures spécifiques à la surface d'une feuille ou des formes simples correspondant à des taches.

- **Étape 3** : Les Étapes Finales (Blocs 4 et 5 avec 256 et 512 filtres)

Aux dernières étapes du processus, la représentation de l'image est très petite sur le plan spatial, mais très profonde en termes de caractéristiques. Les derniers blocs, dotés d'un grand nombre de filtres, réalisent une synthèse finale. Ils sont capables de reconnaître des concepts abstraits qui représentent la signature visuelle d'une maladie. La sortie de cette partie est un résumé très dense et riche en sens, prêt à être interprété par la tête de classification.

❖ **La Tête de Classification**

Cette seconde partie du réseau est la composante décisionnelle du modèle, qui correspond aux blocs jaunes (Flatten), verts (Dense) et gris (Dropout) du schéma. Elle prend en charge les cartes de caractéristiques abstraites, fournies par le corps convolutif, et a pour rôle d'effectuer la classification finale en se basant sur ces informations de haut niveau. Sa structure est la suivante :

- Une première couche de régularisation Dropout avec un taux de 0.25 est appliquée à ce stade.
- Vectorisation des caractéristiques par la couche Flatten :

Cette couche agit comme un pont indispensable entre le corps d'extraction de caractéristiques (convolutif) et la tête de classification (dense). Son unique rôle est de prendre la structure matricielle (hauteur x largeur x canaux) en sortie du dernier bloc de pooling et de l'aplatir en un long vecteur unidimensionnel. Cette transformation est une étape obligatoire car les couches denses, par nature, ne peuvent traiter qu'un vecteur de données en entrée.

- Couche dense intermédiaire :

Nous avons ensuite mis en place une couche Dense entièrement connectée, contenant 1024 neurones et utilisant une activation relue. Le choix de 1024 neurones est un hyperparamètre qui offre une grande capacité d'apprentissage. Chacun de ces neurones est connecté à l'ensemble des caractéristiques du vecteur d'entrée. C'est à ce niveau que le modèle cesse de raisonner localement (sur des parties de l'image) pour commencer à raisonner globalement. Il apprend à identifier des

combinaisons et des corrélations entre toutes les caractéristiques abstraites détectées précédemment pour former des motifs discriminants de haut niveau, qui seront la base de la décision finale.

- Une seconde couche de régularisation Dropout : avec un taux plus élevé de 0.4
- Couche de sortie et décision finale :

Enfin, l'architecture se termine par une couche de sortie Dense. Le nombre de neurones dans cette couche, 38, n'a pas été choisi au hasard : il correspond exactement au nombre de classes distinctes (38) présentes dans le jeu de données utilisé pour ce projet.Chacun de ces 38 neurones produit un score brut (un "logit") qui représente la "preuve" que l'image d'entrée appartient à une classe particulière. Ces scores sont ensuite passés à travers la fonction d'activation softmax. Celle-ci a pour rôle de les normaliser pour les transformer en une distribution de probabilités. Ainsi, la sortie finale de notre modèle est un vecteur de 38 valeurs, chacune représentant la probabilité (entre 0 et 1) que l'image appartienne à une classe, et dont la somme totale est égale à 1. La classe avec la plus haute probabilité est alors retenue comme la prédiction du modèle.

2.4.2. Visualisation et résumé du modèle

Afin d'avoir une vision claire et synthétique de la structure du modèle CNN développé, cette section présente un résumé automatique de l'architecture à l'aide de la commande model.summary() fournie par l'API Keras. Ce résumé permet d'identifier la composition exacte du réseau, les dimensions de sortie à chaque étape, ainsi que le nombre de paramètres à entraîner.

La Figure 2.3 résume toutes les couches du modèle, dans l'ordre de leur exécution. Il indique pour chaque couche :

- Son type (Conv2D, Activation, MaxPooling2D, etc.),
- Sa forme de sortie (Output Shape),
- Et le nombre de paramètres associés.

Layer (type)	Output Shape	Param #
conv2d_38 (Conv2D)	(None, 126, 126, 32)	896
activation_37 (Activation)	(None, 126, 126, 32)	0
max_pooling2d_37 (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_39 (Conv2D)	(None, 61, 61, 64)	18496
activation_38 (Activation)	(None, 61, 61, 64)	0
max_pooling2d_38 (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_40 (Conv2D)	(None, 28, 28, 128)	73856
activation_39 (Activation)	(None, 28, 28, 128)	0
max_pooling2d_39 (MaxPooling2D)	(None, 14, 14, 128)	0
conv2d_41 (Conv2D)	(None, 12, 12, 256)	295168
activation_40 (Activation)	(None, 12, 12, 256)	0
max_pooling2d_40 (MaxPooling2D)	(None, 6, 6, 256)	0
conv2d_42 (Conv2D)	(None, 4, 4, 512)	1180160
activation_41 (Activation)	(None, 4, 4, 512)	0
max_pooling2d_41 (MaxPooling2D)	(None, 2, 2, 512)	0
dropout_27 (Dropout)	(None, 2, 2, 512)	0
flatten_8 (Flatten)	(None, 2048)	0
dense_12 (Dense)	(None, 1024)	2098176
dropout_28 (Dropout)	(None, 1024)	0
dense_13 (Dense)	(None, 38)	38950

Total params: 3,705,702
Trainable params: 3,705,702
Non-trainable params: 0

Figure 2.3 : Structure détaillée de l'architecture du model CNN

Le modèle totalise 20 couches fonctionnelles, structurées comme suit :

- 5 couches convolutionnelles (Conv2D) chacune suivie d'une fonction d'activation ReLU (Activation), et d'un sous-échantillonnage spatial via une couche MaxPooling2D.
- 2 couches de régularisation Dropout.
- 1 couche Flatten.
- 2 couches entièrement connectées Dense (dont la couche de sortie).

Le nombre total de paramètres entraînables est de 3 705 702, ce qui correspond à une architecture modérément profonde, mais suffisamment expressive pour des tâches de classification d'images complexes comme celles rencontrées dans la détection de maladies des plantes.

On observe que la majorité des paramètres est concentrée dans la **couche Dense de 1024 neurones**, qui contient à elle seule 2 098 176 paramètres. Cela s'explique par le fait qu'elle reçoit en entrée un vecteur de 2048 valeurs, produit par la couche Flatten. Ce chiffre (2048) provient du produit des dimensions du tenseur en sortie du dernier bloc convolutionnel, qui a la forme :
(2, 2, 512) Ce qui donne : 2 × 2 × 512 = **2048 neurones**.
Les couches convolutionnelles, bien qu'importantes pour l'extraction des caractéristiques, représentent un poids paramétrique bien moindre. À titre d'exemple, la dernière couche Conv2D à 512 filtres contient 1180160 paramètres, ce qui est déjà significatif mais reste inférieur à la couche Dense.

Ce résumé du modèle permet de confirmer la cohérence structurelle de l'architecture conçue : une progression claire de l'extraction de caractéristiques vers la classification finale, avec une régularisation adaptée et un nombre de paramètres optimisé.

2.4.3. Justification des choix d'activation

Dans le cadre de la conception de notre modèle CNN, nous avons été amenés à faire des choix précis concernant les fonctions d'activation. Celles-ci sont essentielles pour introduire de la non-linéarité dans le réseau, ce qui permet d'apprendre des relations complexes entre les caractéristiques extraites et les classes de sortie. Nous avons opté pour l'utilisation de deux fonctions d'activation distinctes selon leur position dans l'architecture : ReLU pour l'ensemble des couches internes et Softmax pour la couche de sortie.

Nous avons utilisé **ReLU,** définie par :

$$\boldsymbol{f(x) = \max(0, x)}[32]$$

Cette fonction nous a paru la plus adaptée pour plusieurs raisons :

- Simplicité computationnelle : Elle est simple à implémenter et très efficace sur le plan computationnel ce qui nous a permis de réduire le temps d'entraînement.
- Réduction du problème de gradient : Surtout, elle réduit le risque de disparition du gradient, un problème rencontré avec d'anciennes fonctions comme sigmoid ou tanh. Grâce à cela, nous avons pu entraîner un réseau relativement profond sans difficulté majeure.

Ces avantages ont guidé notre décision de l'utiliser après chaque couche convolutionnelle, ainsi qu'au niveau de la couche Dense de 1024 neurones, avant la classification finale.

Nous avons utilisé **Softmax** comme La fonction d'activation en sortie, définie comme suit pour un vecteur de sortie z :

$$\mathbf{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \text{ [33]}$$

Où k est le nombre de classes, ici 38.

Cette fonction a pour effet de transformer le vecteur brut de scores (logits) en probabilités normalisées, telles que :

- Chaque sortie est comprise entre 0 et 1
- La somme de toutes les sorties vaut exactement 1

Cela permet d'interpréter directement la sortie du modèle comme une distribution de probabilité sur les classes possibles. La classe prédite correspond à celle ayant la probabilité la plus élevée. L'utilisation de Softmax est particulièrement appropriée ici, car la tâche est une **classification multi-classes exclusive** : une seule classe correcte par image.

En résumé, nous avons choisi ReLU pour sa rapidité, sa stabilité et sa capacité à bien gérer la profondeur du réseau. Quant à Softmax, il s'impose naturellement comme le choix le plus logique et rigoureux pour une classification exclusive multi-classes. Ces deux fonctions se complètent parfaitement dans le cadre de notre modèle

2.4.4. Fonction de perte et choix de l'optimiseur

Lors de la phase de conception de notre modèle, il a été nécessaire d'anticiper les mécanismes qui guideront l'ajustement des paramètres internes du réseau au cours de son utilisation. Dans cette optique, nous avons identifié la fonction de perte et l'optimiseur les plus adaptés à la nature de notre problème, afin d'assurer la compatibilité avec l'architecture construite et les objectifs visés.

❖ **Notre choix de la fonction de perte :** categorical_crossentropy

Le choix de la fonction de perte s'est porté sur la categorical_crossentropy (entropie croisée catégorielle), car elle est spécifiquement conçue pour les problèmes de classification multi-classes comme le nôtre. En effet, notre objectif est d'assigner chaque image de feuille à une seule étiquette parmi les différentes classes possibles.

Cette fonction mesure l'écart entre la distribution de probabilité prédite par le modèle (et la distribution. Plus cette différence est faible, plus le modèle est précis.

$$L = -\sum_{k=1}^{k} y_k \log(p_k) \quad [34]$$

Ou :

- y_k est la probabilité réelle de la classe k (c'est-à-dire 1 pour la classe correcte, et 0 pour toutes les autres).
- p_k est la probabilité, prédite par notre modèle via la couche Softmax, que l'image appartienne à cette même classe k

❖ **Notre choix de l'optimiseur :** Adam

Pour l'optimisation des poids du réseau, nous avons sélectionné l'optimiseur Adam. Cet algorithme, dont l'acronyme signifie Adaptive Moment Estimation, ajuste automatiquement le taux d'apprentissage pour chaque paramètre d'un modèle en se basant sur les gradients passés[35] .

Ce choix repose sur plusieurs avantages pratiques et théoriques que nous avons jugés décisifs :

- Il combine les bénéfices de deux autres optimisateurs bien connus : AdaGrad (adaptation du learning rate) et RMSprop (prise en compte du moment).
- Il nécessite peu d'ajustements manuels des hyperparamètres par défaut (learning rate, bêta1, bêta2), ce qui est un avantage important dans le cadre d'un projet à contrainte de temps.

En résumé, nous avons choisi d'associer la fonction de perte categorical_crossentropy à l'optimiseur Adam, car cette combinaison est largement éprouvée dans les tâches de classification multi-classes, notamment dans le domaine de la vision par ordinateur.

2.4.5. Stratégies de régularisation

Dans le cadre de la conception de notre modèle, nous avons pris en compte les risques potentiels de surapprentissage, notamment en raison du nombre important de paramètres dans les couches entièrement connectées et de la profondeur globale du réseau. Pour atténuer ces effets, nous avons intégré une stratégie de régularisation simple, efficace et compatible avec notre architecture : le Dropout.

Le Dropout est une méthode de régularisation qui prévient le surapprentissage. Elle consiste à ignorer de manière aléatoire une partie des neurones à chaque étape de l'entraînement, forçant ainsi le réseau à développer des caractéristiques plus robustes et à mieux généraliser[36] .

❖ **Notre choix du Dropout**

Dans le but de prévenir le surapprentissage, nous avons intégré deux couches de régularisation Dropout à des endroits stratégiques de notre architecture.

- La première a été placée après le dernier bloc convolutionnel, avec un taux de 25 %.
- La deuxième, avec un taux de 40 %, se trouve juste après la couche Dense de 1024 neurones.

Pourquoi 25 % après les convolutions ?

Nous avons opté pour un taux modéré (0.25) à ce niveau car :

- Les couches convolutionnelles ont généralement moins de paramètres que les couches denses.
- Un taux trop élevé aurait pu nuire à la stabilité de l'apprentissage en supprimant trop d'informations importantes extraites des images.

Pourquoi 40 % après la couche Dense ?

Le taux plus élevé de 40 % a été choisi pour la couche Dense car :

- C'est à ce niveau que le modèle comporte le plus de paramètres (plus de 2 millions).
- Cette couche joue un rôle critique dans la prise de décision finale, en combinant toutes les caractéristiques extraites.
- En appliquant un Dropout plus agressif ici, nous réduisons fortement le risque que le réseau mémorise les données d'entraînement.

❖ **Alternatives considérées**

D'autres méthodes de régularisation, comme l'arrêt précoce (Early Stopping) ou la pénalisation L2 des poids (Weight Decay), ont été envisagées mais non intégrées à ce stade. Notre objectif était de maintenir l'architecture aussi simple que possible, tout en assurant une bonne stabilité conceptuelle.

Le recours au Dropout s'est inscrit naturellement dans notre démarche, en tant que solution de régularisation légère et adaptée qui renforce la robustesse du réseau sans en complexifier l'architecture. Son application a suivi une logique adaptative : les valeurs (0.25 et 0.4) ont été choisies pour moduler l'intensité du Dropout en fonction de la densité de chaque partie du réseau, le renforçant là où les connexions sont plus nombreuses et le risque de surapprentissage plus critique.

2.5. Conclusion

Ce chapitre a établi l'architecture complète de notre solution, en se concentrant sur la conception d'un modèle de réseau de neurones convolutifs (CNN) sur mesure pour la détection des maladies des plantes. Nous avons défini un processus de prétraitement simple, redimensionnant les images à 128×128 pixels, tout en justifiant l'absence de normalisation et d'augmentation de données à ce stade.

Le cœur de ce chapitre a été consacré à la construction d'un modèle CNN séquentiel (from scratch), ce modèle se compose d'un corps d'extraction de caractéristiques à cinq blocs et d'une tête de classification décisionnelle. Des choix techniques clés, comme l'utilisation des activations ReLU/Softmax, de l'optimiseur Adam et de la régularisation Dropout, ont été faits pour assurer un apprentissage stable et efficace.

Cette conception détaillée et justifiée constitue le fondement de notre modèle, dont l'entraînement et l'évaluation des performances feront l'objet du chapitre suivant.

III

Implémentation et résultats

3.1. Introduction

Après avoir détaillé la conception de notre modèle convolutionnel dans le chapitre précédent nous présentons ici la phase d'implémentation et d'expérimentation, qui constitue une étape clé du projet. Il s'agit de mettre en œuvre concrètement l'architecture définie, de l'entraîner sur des données réelles, puis d'évaluer sa capacité à accomplir la tâche de classification des maladies des plantes à partir d'images.

3.2. Environnement de Développement

La mise en œuvre de notre modèle de réseau de neurones convolutif et la conduite des expérimentations ont nécessité un environnement technique spécifique. Cette section détaille les composantes matérielles et logicielles, les librairies ainsi que l'environnement d'exécution qui ont servi de fondation à notre travail.

3.2.1. Spécifications Matérielles et Logicielles

L'entraînement en apprentissage profond est une tâche intensive dont la performance dépend de la configuration utilisée. Pour garantir la transparence et la reproductibilité, les spécifications de la machine d'entraînement sont détaillées dans le tableau ci-dessous.

Composant	Spécification
Processeur (CPU)	Intel Core i7-8550U
Mémoire vive (RAM)	8 Go DDR4
Processeur Graphique (GPU)	NVIDIA GeForce MX110
Système d'exploitation	Windows 10

Tableau 3.1 : Spécifications matérielles et logicielles

3.2.2. Langages, Librairies

Le développement de la solution s'est appuyé sur l'écosystème Python et un ensemble de librairies open source reconnues dans le domaine de la science des données et de l'apprentissage profond :

- **Python** :

Python est un langage de programmation polyvalent, utilisé pour le développement web, la science des données et le machine learning. Gratuit et multiplateforme, il s'intègre à tous les systèmes pour un développement accéléré [37].

Nous Avons choisi Python pour ses nombreuses bibliothèques spécialisées en traitement d'images et deep learning.

- **Tensorflow :**

TensorFlow est une librairie open-source qui permet aux développeurs de construire et déployer des applications d'apprentissage automatique plus rapidement. En tant que bibliothèque de calcul numérique, elle prend en charge l'ensemble du processus, depuis la préparation des données et la construction du modèle jusqu'à la diffusion des prédictions à grande échelle et leur optimisation continue [38].

- **Keras :**

Conçue pour la simplicité et la vitesse, Keras est l'API de haut niveau officielle et intégrée de TensorFlow. Elle fournit des composants modulaires (couches, optimiseurs) pour assembler et entraîner rapidement des modèles de deep learning. Cette approche masque la complexité des calculs, ce qui permet au développeur de se concentrer sur l'architecture et l'expérimentation [39].

- **NumPy** :

c'est une bibliothèque open-source fondamentale pour le calcul scientifique en langage Python. C'est un outil puissant et indispensable dans les domaines de la science des données, de l'ingénierie et des

mathématiques, où il est utilisé pour réaliser des calculs complexes, notamment sur les matrices et les tableaux multidimensionnels [40].

- **Scikit-learn :**

Scikit-learn est une librairie open-source spécialisée dans la machine learning. Elle propose des fonctionnalités performantes pour explorer, analyser et modéliser les données. Construite sur des bases solides comme NumPy, SciPy et Matplotlib, elle prend en charge de nombreuses techniques telles que la classification, la régression, le clustering et la réduction de la dimensionnalité [41].

- **Matplotlib :**

Matplotlib est une bibliothèque puissante de Python utilisée pour créer des visualisations de données statiques, interactives et dynamiques, comme des graphiques en barres, des histogrammes ou des diagrammes en secteurs. Très répandue en science des données, elle permet de transformer efficacement les données en représentations graphiques claires et personnalisables [42].

3.2.3. Environnement d'Exécution

Le cycle de développement de notre travail, allant de l'expérimentation à l'analyse des résultats, a été entièrement mené au sein de l'environnement de calcul interactif **Jupyter Notebook**. Cet outil a été privilégié pour sa flexibilité et sa capacité à unifier les différentes phases du travail en science des données.

Il nous a permis de combiner dans un seul et même document :

- **Le code source Python** pour la construction et l'entraînement du modèle.
- **L'exécution de code par cellules**, ce qui a grandement facilité les phases de test et de débogage de chaque partie du processus.
- **La visualisation instantanée des résultats**, comme l'affichage des courbes de performance du modèle (notamment la précision et la perte).

Les expérimentations ont été menées dans un environnement Jupyter Notebook fonctionnant localement sur la machine et le système d'exploitation décrits précédemment, en s'appuyant sur les librairies et frameworks mentionnés.

3.3. Préparation du Jeu de Données

Après avoir configuré notre environnement, l'étape suivante est la préparation des données d'entrée. Cette phase est fondamentale, car la qualité et la structure des données utilisées pour l'entraînement déterminent directement la performance et la fiabilité de notre modèle de détection.

3.3.1. Description du Jeu de Données

Pour entraîner notre modèle de détection des maladies des plantes, nous avons utilisé le jeu de données "New Plant Diseases Dataset" [43]. Ce jeu de données contient un grand nombre d'images de feuilles agricoles, classées selon leur état de santé (saines ou malades) et l'espèce végétale concernée.

Le dataset est organisé de manière structurée, chaque classe de maladie étant représentée par un dossier spécifique contenant des images d'exemples. Au total, la version utilisée dans ce projet comprend 38 classes distinctes, correspondant à différentes maladies (fongiques, bactériennes, etc) et à différentes plantes cultivées (tomate, maïs, pomme de terre, etc).

Les images sont :

- De format .jpg
- En couleur (RGB)
- De bonne qualité
- Et prise sur fond clair, avec un éclairage homogène.

La figure ci-dessous représente quelques exemples d'images issues de ce dataset :

Figure 3.1 : Exemples de Cedar-apple rust sur feuilles de pommier.

3.3.2. Répartition des données

Un avantage majeur du jeu de données [43] est qu'il est fourni avec une segmentation pré-établie, ce qui nous évite d'avoir à effectuer une division manuelle et garantit une base d'expérimentation standard et reproductible. La structure des dossiers est déjà organisée pour l'entraînement et l'évaluation.

La répartition est la suivante :

- **Ensemble d'entraînement (train)** : Ce répertoire contient 80% du jeu de données. Il est exclusivement destiné à la phase d'apprentissage de notre modèle.
- **Ensemble de validation (valid)** : Ce répertoire contient les 20% restants. Il sert à évaluer la performance du modèle à la fin de chaque époque durant l'entraînement, afin de surveiller sa capacité de généralisation.
- **Ensemble de test (test)** : Un troisième répertoire, plus petit, est également fourni. Il contient un ensemble d'images de test complètement distinctes, destinées à une évaluation finale et ponctuelle du modèle après l'entraînement.

3.3.3. Chargement des Données

Le chargement d'un jeu de données de près de 87 000 images représente un défi technique majeur en termes d'utilisation de la mémoire (RAM). Une approche naïve qui consisterait à tout charger en mémoire simultanément la saturerait instantanément, pour contourner ce problème nous avons utilisé image_dataset_from_directory [44]. Cette fonction crée un flux de données qui charge et prétraite les images par lots à la demande, réduisant l'utilisation mémoire.

Les extraits de code suivants illustrent le chargement des ensembles d'entraînement et de validation avec le nombre d'images traitées pour chaque ensemble.

```
training_set = tf.keras.utils.image_dataset_from_directory(
    'train',
    labels="inferred",
    label_mode="categorical",
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
)
```

```
Found 70295 files belonging to 38 classes.
```

```
validation_set = tf.keras.utils.image_dataset_from_directory(
    'valid',
    labels="inferred",
    label_mode="categorical",
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
)
```

```
Found 17572 files belonging to 38 classes.
```

Comme le confirment les sorties de code, notre jeu de données est donc précisément composé de 70 295 images pour l'entraînement et 17 572 images pour la validation. La fonction a correctement identifié les 38 classes distinctes dans les deux ensembles.

Ces pipelines de données sont maintenant prêts à être fournis au modèle, avec des lots (batch_size) de 32 images et une taille d'image de 128x128 pixels, conformément à notre configuration.

3.4. Processus d'Entraînement du Modèle

Dans cette section, nous allons décrire en détail comment le modèle, que nous avons conçu au Chapitre 2, a été entraîné avec les données que nous venons de préparer. C'est ici que nous expliquons le processus d'apprentissage.

3.4.1. Configuration des Hyperparamètres d'Entraînement

Le succès du processus d'apprentissage ne dépend pas uniquement de l'architecture du modèle, mais aussi d'un ensemble de paramètres configurés avant le lancement, appelés hyperparamètres. Ils contrôlent la manière dont le modèle apprend. Nous avons porté une attention particulière à la configuration de trois hyperparamètres clés : la taille des lots, le nombre d'époques et le taux d'apprentissage de l'optimiseur.

- **Taille de lot (batch size)**

Cet hyperparamètre correspond au nombre d'images que le modèle analyse avant de procéder à une mise à jour de ses poids. Nous avons retenu une taille de lot de 32, une valeur standard largement utilisée dans les réseaux convolutionnels. Elle offre un bon compromis entre vitesse de calcul, stabilité numérique et efficacité mémoire. Cette valeur a été définie dès le chargement des données avec image_dataset_from_directory ().

- **Nombre d'époques (epochs)**

Cet hyperparamètre détermine combien de fois le modèle va parcourir l'intégralité du jeu de données d'entraînement.

Nous avons fixé le nombre total d'époques à 10. Ce choix est un standard en apprentissage profond car il représente le meilleur compromis : il est assez grand pour fournir une estimation stable de l'erreur et optimiser la vitesse de calcul, tout en étant assez petit pour garantir une utilisation efficace de la mémoire et aider le modèle à mieux généraliser.

- **Taux d'apprentissage (learning_rate)**

Le taux d'apprentissage détermine l'ampleur des ajustements effectués par l'optimiseur après chaque mise à jour. Un taux trop faible ralentit fortement l'entraînement, tandis qu'un taux trop élevé peut rendre l'apprentissage instable, voire le faire diverger.

Nous avons choisi un taux de 0.0001, dix fois plus faible que la valeur par défaut (0.001).Ce réglage permet d'éviter les instabilités pendant l'entraînement, surtout dans les couches profondes où les gradients peuvent varier de manière importante. Il évite les sauts brutaux dans la fonction de perte et assure une convergence progressive. Particulièrement adapté à notre CNN complexe, ce choix optimise l'apprentissage des nombreux paramètres.

3.4.2. Compilation et Entraînement du modèle

Après avoir défini l'architecture du modèle et configuré ses hyperparamètres, l'étape finale de l'implémentation consiste à compiler le modèle puis à lancer le processus d'apprentissage. Ces deux actions sont réalisées via des commandes spécifiques de l'API Keras.

1. **La Compilation du Modèle**

La compilation est l'étape qui prépare le modèle à l'entraînement. Elle consiste à lui attacher un optimiseur, une fonction de perte et des métriques de performance à suivre. Conformément aux choix de conception établis au chapitre précédent, nous avons configuré le modèle avec :

- **L'optimiseur** Adam, avec le taux d'apprentissage personnalisé de 0.0001 que nous avons fixé.
- **La fonction de perte** categorical_crossentropy, qui est spécifiquement conçue pour les problèmes de classification multi-classes comme le nôtre.
- **La métrique** accuracy (précision) pour évaluer la performance du modèle après chaque époque.

Le code d'implémentation correspondant est le suivant :

```
model.compile(optimizer=tf.keras.optimizers.Adam(
    learning_rate=0.0001),loss='categorical_crossentropy',metrics=['accuracy'])
```

2. Entraînement du modèle

La phase d'entraînement démarre après compilation du modèle en appelant la méthode fit(). Cette commande gère l'entraînement : elle prend en entrée les ensembles de données d'entraînement et de validation que nous avons préparés, et exécute l'apprentissage sur le nombre d'époques défini (10 dans notre cas).

Voici un extrait représentatif du code utilisé :

```
model_training = model.fit(x=training_set,validation_data=validation_set,epochs=10)
```

L'objet model_training stocke l'évolution de plusieurs métriques (accuracy, loss) sur chaque époque, permettant une analyse visuelle ultérieure à l'aide de graphiques.

La figure ci-dessous est une capture d'écran du log complet de notre session d'entraînement. Elle constitue une trace concrète de l'exécution de notre entraînement et permet de suivre l'évolution des métriques de performance, époque par époque.

```
Epoch 1/10
2197/2197 [==============================] - 1521s 692ms/step - loss: 1.4535 - accuracy: 0.5895 - val_loss: 0.4811 - val_accuracy: 0.8494
Epoch 2/10
2197/2197 [==============================] - 1494s 680ms/step - loss: 0.5018 - accuracy: 0.8421 - val_loss: 0.2939 - val_accuracy: 0.9084
Epoch 3/10
2197/2197 [==============================] - 1551s 706ms/step - loss: 0.3049 - accuracy: 0.9019 - val_loss: 0.2495 - val_accuracy: 0.9227
Epoch 4/10
2197/2197 [==============================] - 1508s 686ms/step - loss: 0.2065 - accuracy: 0.9313 - val_loss: 0.1417 - val_accuracy: 0.9542
Epoch 5/10
2197/2197 [==============================] - 1506s 686ms/step - loss: 0.1523 - accuracy: 0.9496 - val_loss: 0.1342 - val_accuracy: 0.9561
Epoch 6/10
2197/2197 [==============================] - 1518s 691ms/step - loss: 0.1177 - accuracy: 0.9611 - val_loss: 0.1264 - val_accuracy: 0.9597
Epoch 7/10
2197/2197 [==============================] - 1492s 679ms/step - loss: 0.0991 - accuracy: 0.9672 - val_loss: 0.1163 - val_accuracy: 0.9621
Epoch 8/10
2197/2197 [==============================] - 1493s 680ms/step - loss: 0.0796 - accuracy: 0.9736 - val_loss: 0.0891 - val_accuracy: 0.9723
Epoch 9/10
2197/2197 [==============================] - 1493s 679ms/step - loss: 0.0683 - accuracy: 0.9777 - val_loss: 0.0890 - val_accuracy: 0.9724
Epoch 10/10
2197/2197 [==============================] - 1540s 701ms/step - loss: 0.0600 - accuracy: 0.9801 - val_loss: 0.1152 - val_accuracy: 0.9660
```

Figure 3.2 : Log complet de l'entraînement du modèle sur 10 époques.

Le log confirme l'exécution complète des 10 époques d'entraînement. Avec les poids désormais optimisés, le modèle est prêt pour l'évaluation. L'analyse quantitative et l'interprétation des performances visibles dans ce log feront l'objet de la section suivante.

3.5. Évaluation et Analyse des Performances

Maintenant que le modèle est entraîné, cette section est consacrée à l'évaluation objective de sa performance. Nous y mènerons une analyse approfondie pour quantifier sa précision, visualiser son comportement d'apprentissage.Cette évaluation s'appuiera sur l'analyse des courbes d'apprentissage, la mesure des performances sur l'ensemble de test et l'interprétation de la matrice de confusion.

3.5.1. Métriques d'Évaluation

L'évaluation de la performance de notre modèle de classification s'appuie sur plusieurs métriques couramment utilisées en apprentissage profond. Chacune de ces métriques offre une perspective différente sur la qualité des prédictions du modèle.

- **Perte (loss) :**

La perte, et plus spécifiquement l'entropie croisée catégorielle (categorical_crossentropy) que nous avons choisie, mesure à quel point les prédictions du modèle sont "éloignées" de la vérité[34]. Une valeur de perte faible indique que le modèle est très confiant et correct dans ses prédictions. C'est la métrique que le modèle cherche à minimiser durant son entraînement.

- **La justesse (Accuracy) :**

La justesse (accuracy) mesure la proportion de prédictions correctes (positives et négatives) sur l'ensemble des prédictions. Elle est utile quand les classes sont équilibrées, mais devient trompeuse avec des données déséquilibrées. Dans ce cas, d'autres métriques comme le rappel ou la précision sont souvent préférées.

$$\text{Accuracy} = \frac{\text{TP+T}}{\text{TP+FP+TN+F}} \text{ [45]}$$

- **Précision (Precision):**

La précision représente la proportion des prédictions positives qui sont réellement correctes. Elle est utile pour évaluer les performances d'un modèle lorsqu'il est crucial de minimiser les faux positifs. Toutefois, dans les ensembles de données très déséquilibrés, elle devient moins fiable et présente souvent une relation inverse avec le rappel.

$$\text{Precision} = \frac{\text{TP}}{\text{TP+}} \text{[45]}$$

- **Rappel (Recall):**

Métrique qui mesure la capacité d'un modèle à détecter toutes les instances positives réelles (taux de vrais positifs).

$$\text{Recall} = \frac{TP}{TP+FN} \text{ [45]}$$

- **F1-score :**

Moyenne harmonique entre la précision et le rappel, Cette moyenne harmonique, moins sensible aux valeurs extrêmes, offre une évaluation équilibrée, avec un score maximal de 1 indiquant une performance optimale

$$\text{F1-score} = 2 * \frac{\text{Précision} * \text{Rappel}}{\text{Précision} + \text{Rappel}} \text{ [46]}$$

- **Matrice de confusion:**

La matrice de confusion permet d'évaluer un modèle de classification en comparant ses prédictions aux étiquettes réelles. Elle aide à identifier les points forts et faibles du modèle [47].

La matrice se compose de lignes représentant les prédictions du modèle (positif ou négatif) et de colonnes représentant la vérité terrain (ce que les étiquettes indiquent réellement).

Les cellules internes correspondent à des comptages d'occurrences, répartis comme suit :

- Vrais positifs (TP) : cas où le modèle a correctement identifié une instance positive.
- Faux positifs (FP) : cas où le modèle a incorrectement prédit une instance négative comme étant positive.
- Faux négatifs (FN) : cas où le modèle a omis de détecter une instance positive.
- Vrais négatifs (TN) : cas où le modèle a correctement prédit une instance négative.

La figure ci-dessous illustre visuellement cette structure pour un cas de classification binaire.

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Figure 3.3 : Structure d'une matrice de confusion binaire [47]

3.5.2. Analyse des Courbes d'Apprentissage

L'analyse des courbes d'apprentissage est une méthode visuelle puissante pour diagnostiquer le comportement de notre modèle durant les 10 époques de son entraînement. En traçant l'évolution de l'accuracy et de la perte, nous pouvons évaluer la qualité de l'apprentissage et détecter d'éventuels problèmes comme le surapprentissage.

- **Évolution de l'Accuracy**

Le premier graphique illustre l'évolution de l'accuracy (la précision globale) sur les ensembles d'entraînement et de validation.Il nous permet de visualiser si le modèle améliore sa capacité à classer correctement les images au fil du temps, et si cette amélioration se généralise bien aux données qu'il n'a pas apprises.

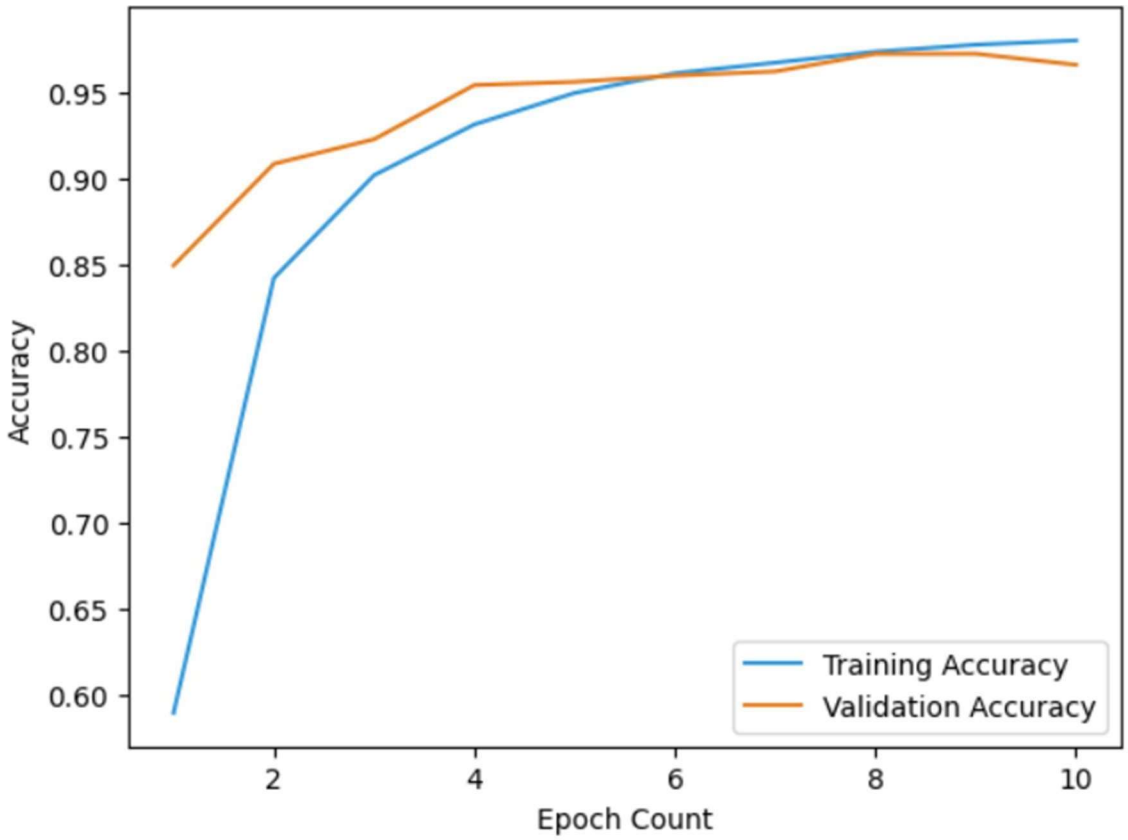

Figure 3.4 : Courbes d'évolution de l'accuracy.

L'analyse de cette figure nous fournit des informations très positives sur la dynamique d'apprentissage de notre modèle :

- **Une Convergence Rapide et Stable** :

La courbe de l'accuracy d'entraînement (en bleu) montre une progression très rapide, partant de moins de 60% pour dépasser les 90% en seulement trois époques, avant de se stabiliser autour de **98%** à la fin de l'entraînement. Cela indique que le modèle apprend très efficacement.

➢ **Une Excellente Généralisation** :

La courbe de l'accuracy de validation (en orange) suit une trajectoire presque identique à celle de l'entraînement. Elle grimpe rapidement pour atteindre un plateau juste au-dessus de 96%. Le fait que les deux courbes soient très proches et ne divergent pas est le signe le plus important : **il n'y a pas de surapprentissage (overfitting) significatif**. Le modèle est donc capable de généraliser ses connaissances à des données qu'il n'a jamais vues durant son apprentissage.

➢ **Stabilité du Modèle** :

Les deux courbes sont lisses et ne présentent pas de fortes oscillations, ce qui confirme que les hyperparamètres choisis, notamment le taux d'apprentissage, ont permis une convergence stable.

En conclusion, ce premier graphique valide la bonne santé de notre processus d'entraînement. Le modèle apprend bien et de manière robuste.

- **Évolution de la Perte (Loss)**

Le second graphique trace l'évolution de la fonction de perte (loss) sur les ensembles d'entraînement et de validation. Cette courbe est en quelque sorte le miroir de celle de l'accuracy : une perte qui diminue signifie que le modèle devient plus "sûr" de ses prédictions et que son erreur globale se réduit.

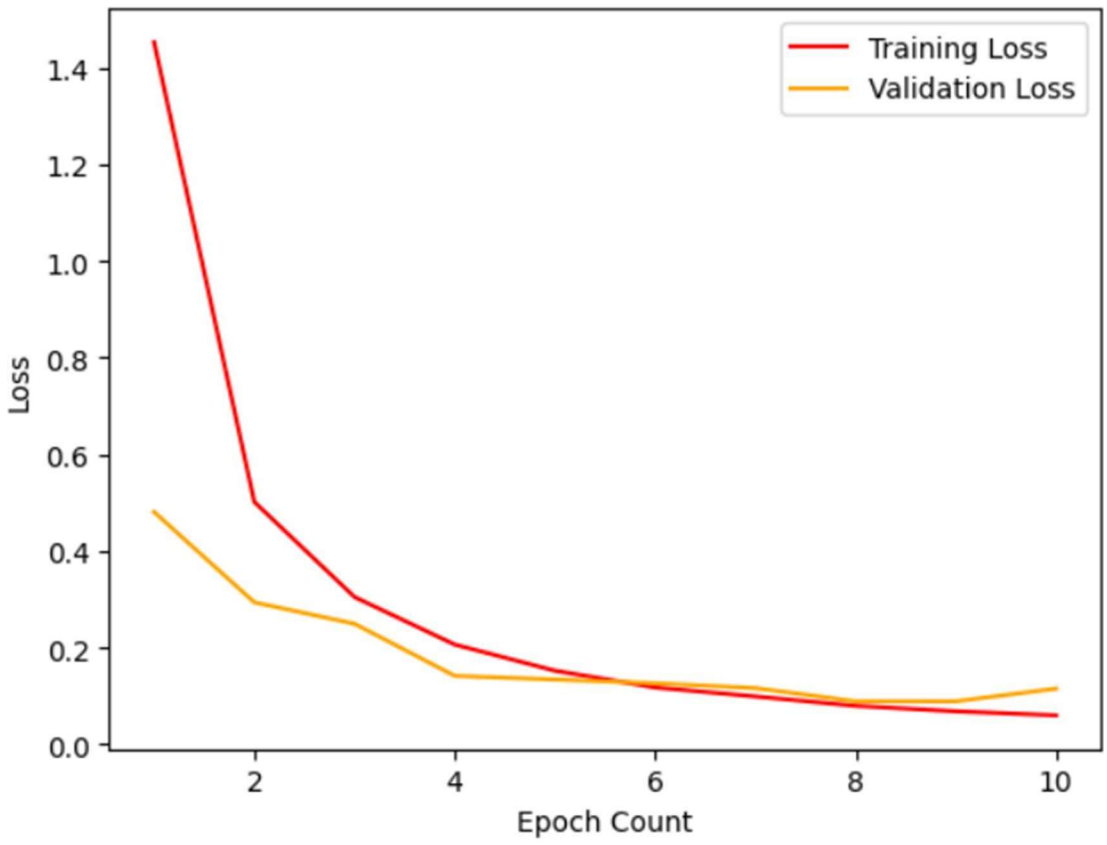

Figure 3.5 : Courbes d'évolution de la perte

L'analyse de ces courbes de perte renforce les conclusions précédentes :

- **Une Diminution Rapide et Convergente :**

La perte d'entraînement (en rouge) décroît fortement, passant d'environ 1.45 en époque 1 à une valeur proche de 0.06 en fin d'apprentissage. Parallèlement, la perte de validation (en orange) diminue aussi très rapidement pour se stabiliser à une valeur très basse, aux alentours de 0.1.

- **Absence de Surapprentissage :**

Les deux courbes restent très proches l'une de l'autre tout au long des 10 époques. Le léger rebond de la courbe de validation à la toute fin est minime et n'indique pas un surapprentissage problématique à ce stade. Cela confirme que le modèle ne mémorise pas les données d'entraînement mais généralise bien.

La double confirmation apportée par l'augmentation de l'accuracy et la diminution de la perte nous permet d'affirmer que le modèle a été entraîné avec succès.

3.5.3. Analyse de la Matrice de Confusion

L'outil principal pour cette analyse détaillée est la matrice de confusion, qui confronte les classes prédites par le modèle aux classes réelles. Une matrice de confusion complète de 38x38 serait cependant trop grande pour être lisible. Pour cette raison, nous avons choisi de visualiser sous forme de carte de chaleur (heatmap) un extrait de cette matrice portant sur les 13 premières classes, comme le montre la figure ci-dessous.

Ce graphique nous montre principalement deux choses :

- **La diagonale principale** (allant du coin supérieur gauche au coin inférieur droit) représente le nombre de prédictions correctes. Une diagonale fortement illuminée, comme c'est le cas sur notre figure, indique un grand nombre de classifications réussies.
- **Les cellules hors diagonale** représentent les erreurs. Une cellule non nulle indique une confusion : le modèle a prédit une classe à la place d'une autre. Des cellules sombres, proches de zéro, sont donc le signe d'un modèle performant qui ne confond que très peu les classes entre elles.

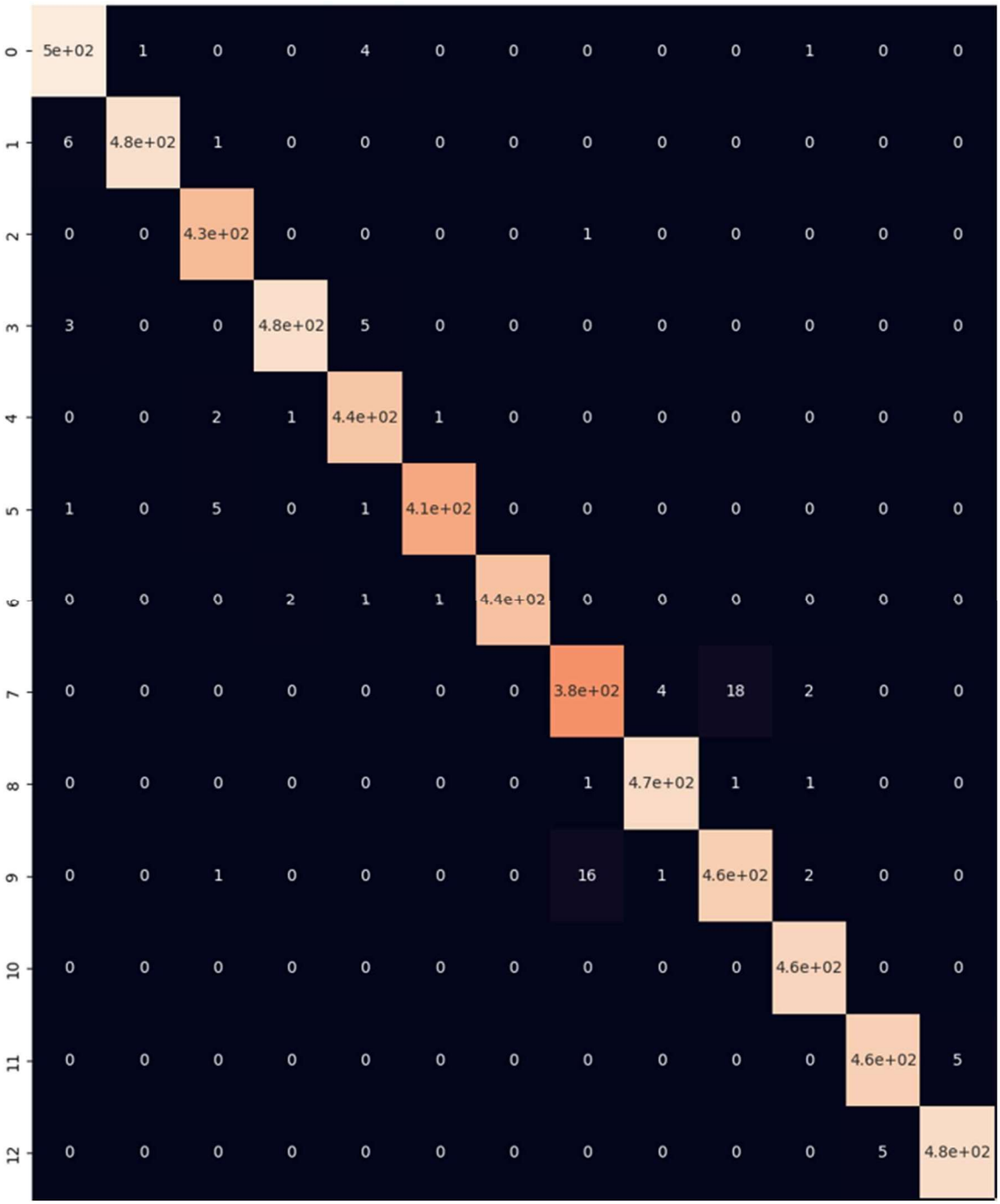

Figure 3.6 : Extrait de la matrice de confusion pour les 13 premières classes.

L'analyse visuelle de la matrice de confusion confirme la très haute performance de notre modèle. On observe une diagonale principale fortement marquée, ce qui indique qu'une grande majorité des images de chaque classe a été correctement classifiée.

En dehors de cette diagonale, la plupart des cellules restent sombres, signifiant que les erreurs de classification sont globalement rares.

3.5.4. Analyse Détaillée des Performances par Classe

Après avoir observé la convergence globale du modèle lors de l'entraînement, et en complément de l'analyse visuelle fournie par la matrice de confusion, il est essentiel d'analyser ses performances de manière plus fine, classe par classe, nous avons utilisé l'ensemble de validation. Idéalement, cette analyse se ferait sur un large ensemble de test, mais celui-ci n'étant pas disponible pour une évaluation statistique robuste, l'ensemble de validation reste notre meilleur indicateur pour comprendre les forces et les faiblesses du modèle dans la distinction des 38 classes.

Le rapport de classification ci-dessous résume quantitativement la performance du modèle. Il présente, pour chaque classe, les métriques clés de précision, rappel et F1-score :

	Précision	Rappel	F1-score	Support
Apple__Apple_scab	0.93	0.98	0.95	504
Apple__Black_rot	0.99	0.97	0.98	497
Apple__Cedar_apple_rust	0.95	0.98	0.97	440
Apple__healthy	0.96	0.95	0.96	502
Blueberry__healthy	0.95	0.98	0.97	454
Cherry(including_sour)___Powdery_mildew	0.98	0.97	0.97	421
Cherry(including_sour)___healthy	1	0.97	0.98	456
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	0.95	0.94	0.94	410
Corn_(maize)___Common_rust_	0.99	0.99	0.99	477
Corn_(maize)___healthy	0.98	1	0.99	465
Corn_(maize)___Northern_Leaf_Blight	0.96	0.96	0.96	477
Grape___Black_rot	0.99	0.97	0.98	472
Grape___Esca_(Black_Measles)	0.99	0.99	0.99	480
Grape___healthy	0.99	0.99	0.99	423
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	0.99	1	0.99	430
Orange___Haunglongbing_(Citrus_greening)	1	0.96	0.98	503
Peach___Bacterial_spot	0.98	0.93	0.96	459
Peach___healthy	0.92	1	0.96	432

Pepper,_bell___Bacterial_spot	0.99	0.94	0.96	478
Pepper,_bell___healthy	0.96	0.95	0.96	497
Potato___Early_blight	0.98	0.99	0.99	485
Potato___healthy	0.98	0.98	0.98	456
Potato___Late_blight	0.97	0.94	0.95	485
Raspberry___healthy	1	0.97	0.98	445
Soybean___healthy	0.98	0.97	0.98	505
Squash___Powdery_mildew	0.97	0.99	0.98	434
Strawberry___healthy	1	1	1	456
Strawberry___Leaf_scorch	1	0.95	0.97	444
Tomato___Bacterial_spot	0.99	0.92	0.96	425
Tomato___Early_blight	0.85	0.95	0.90	480
Tomato___healthy	0.94	0.99	0.97	481
Tomato___Late_blight	0.93	0.92	0.92	463
Tomato___Leaf_Mold	0.97	0.97	0.97	470
Tomato___Septoria_leaf_spot	0.97	0.85	0.91	436
Tomato___Spider_mites Two-spotted_spider_mite	0.95	0.97	0.96	435
Tomato___Target_Spot	0.91	0.91	0.91	457
Tomato___Tomato_mosaic_virus	0.96	1	0.98	448
Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.98	0.99	0.98	490

Tableau 3.2 : Rapport de classification détaillé par classe.

L'analyse de ce rapport révèle une performance globale exceptionnelle de notre modèle sur la quasi-totalité des 38 classes. Plusieurs observations clés peuvent être faites :

1. L'analyse de ce rapport révèle une performance globale exceptionnelle. La grande majorité des 38 classes obtient un F1-score supérieur ou égal à 0.95, ce qui témoigne de la grande robustesse et de la précision du modèle.
2. Bien que la performance globale soit excellente, le rapport identifie les F1-scores les plus bas, bien que toujours élevés, concernent certaines maladies de la tomate, notamment Tomato___Early_blight (0.90) et Tomato___Septoria_leaf_spot (0.91). Cette performance

légèrement inférieure suggère une grande similarité visuelle entre ces maladies, rendant leur distinction plus complexe. C'est donc un axe d'amélioration potentiel pour de futurs travaux.

3.6. Application du Modèle pour le Diagnostic

Après avoir évalué les performances quantitatives de notre modèle, cette section a pour but d'illustrer son application pratique nous allons démontrer le fonctionnement complet du système : depuis la prédiction d'une maladie sur une nouvelle image jusqu'à la présentation d'une recommandation associée.

Le rôle de notre modèle CNN est de fournir un diagnostic précis à partir d'une image. Cependant pour que ce diagnostic soit réellement utile, il doit pouvoir déboucher sur une action ou un conseil. Pour illustrer comment la prédiction du modèle peut être transformée en aide concrète nous avons constitué une base de connaissances qui associe les maladies détectables à des recommandations spécifiques.

Le tableau ci-dessous présente un échantillon de 6 de ces correspondances, qui serviront de référence pour les exemples pratiques qui suivent.

Nom Commun de la Maladie	Recommandation / Conseil
Cedar-apple rust	Retirez et détruisez les feuilles et branches infectées [48].
apple scab	Nettoyer et détruire les feuilles mortes en automne [49].
Corn Common Rust	Appliquer un fongicide foliaire tôt, avant que la maladie ne s'étende trop [50].
EarlyBlight (Potato)	Appliquer des fongicides foliaires de manière préventive [51].
EarlyBlight (tomato)	Pratiquer une rotation des cultures (ne pas replanter tomates, poivrons ou pommes de terre au même endroit pendant 2 ans) [52].
Tomato yellow leaf curl virus	Aucun traitement contre le virus, la seule solution est de contrôler la mouche blanche pour éviter l'infection [53].

Tableau 3.3 : Exemples de diagnostics et recommandations associées.

Maintenant que notre base de connaissances est établie, nous pouvons démontrer le fonctionnement de notre système sur des cas concrets, en utilisant des images de notre ensemble de test que le modèle n'a jamais vues.

Exemple 1 : Diagnostic de la Tavelure du Pommier (Apple scab) :

Pour notre premier cas pratique, nous avons sélectionné l'image de feuille de pommier suivante, issue de notre ensemble de test

Figure 3.7 : Image de test pour la maladie (Apple scab)

Cette image a été fournie à notre modèle, qui a correctement prédit la classe Apple___Apple_scab. Suite à ce diagnostic, notre script interroge le dictionnaire de recommandations pour récupérer le conseil correspondant.

La figure ci-dessous montre le résultat final de cette démonstration, tel qu'il pourrait être présenté à un utilisateur.

Figure 3.8 : Résultat final affichant le diagnostic et la recommandation pour la maladie Apple_scab.

Exemple 2 : Diagnostic de la Brûlure Précoce de la Pomme de Terre (Potato Early blight)

Pour notre second cas pratique, nous avons sélectionné l'image suivante, issue de notre ensemble de test, afin de démontrer la performance du modèle sur une donnée entièrement nouvelle.

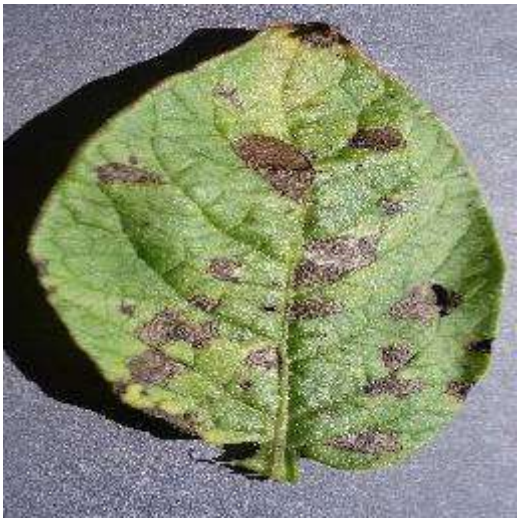

Figure 3.9 : Image de test pour la maladie (Potato Early_blight)

Cette image a été fournie à notre modèle, qui a correctement prédit la classe Potato___Early_blight. Suite à ce diagnostic, notre script interroge le dictionnaire de recommandations pour récupérer le conseil correspondant.

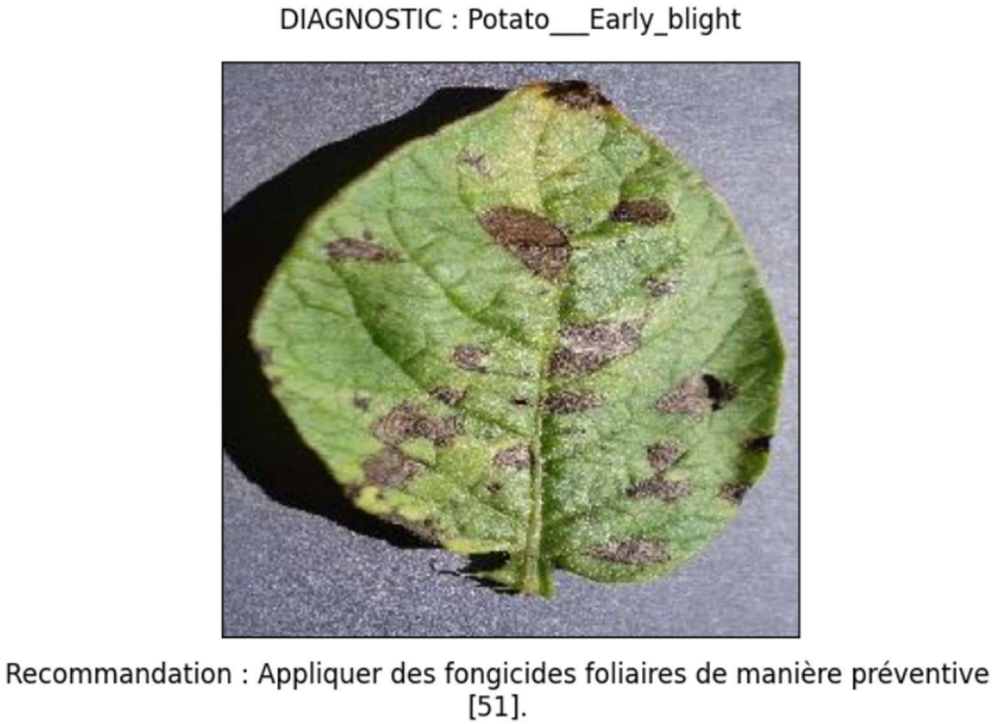

Figure 3.10 : Résultat final affichant le diagnostic et la recommandation pour la maladie (Potato Early blight)

Ces exemples d'application sur des données inédites valident le fonctionnement global de notre projet. Ils confirment la capacité du système à passer d'une simple image à un diagnostic précis et automatisé, accomplissant ainsi l'un des objectifs centraux de ce travail, la performance démontrée ici affirme le potentiel de notre modèle en tant qu'outil d'aide au diagnostic fiable dans le domaine agricole.

- **Déploiement d'une Interface Web Applicative**

Alors que les exemples précédents, réalisés au moyen de scripts dans un environnement Jupyter Notebook, ont démontré avec succès la capacité de notre chaîne de traitement à poser un diagnostic et à formuler une recommandation pertinente, ils constituent avant tout une validation technique de notre approche.

Pour traduire le potentiel de notre modèle en un outil concret, intuitif et directement exploitable par un utilisateur final (tel qu'un agriculteur ou un technicien agricole), nous avons développé une interface web. Cette application encapsule la complexité du modèle de Deep Learning, et de la base de connaissances dans une plateforme simple d'utilisation. L'objectif est de permettre à quiconque sans aucune compétence en programmation d'obtenir un diagnostic fiable en quelques clics.

L'utilisateur n'a qu'à téléverser une image de la feuille suspecte, et le système se charge d'exécuter le modèle, d'interpréter le résultat et d'afficher un diagnostic clair ainsi que la recommandation associée.

Figure 3.11 : Interface de la page "Détection de maladie"

Une fois l'image téléversée, le système analyse la feuille et affiche le résultat final à l'utilisateur :

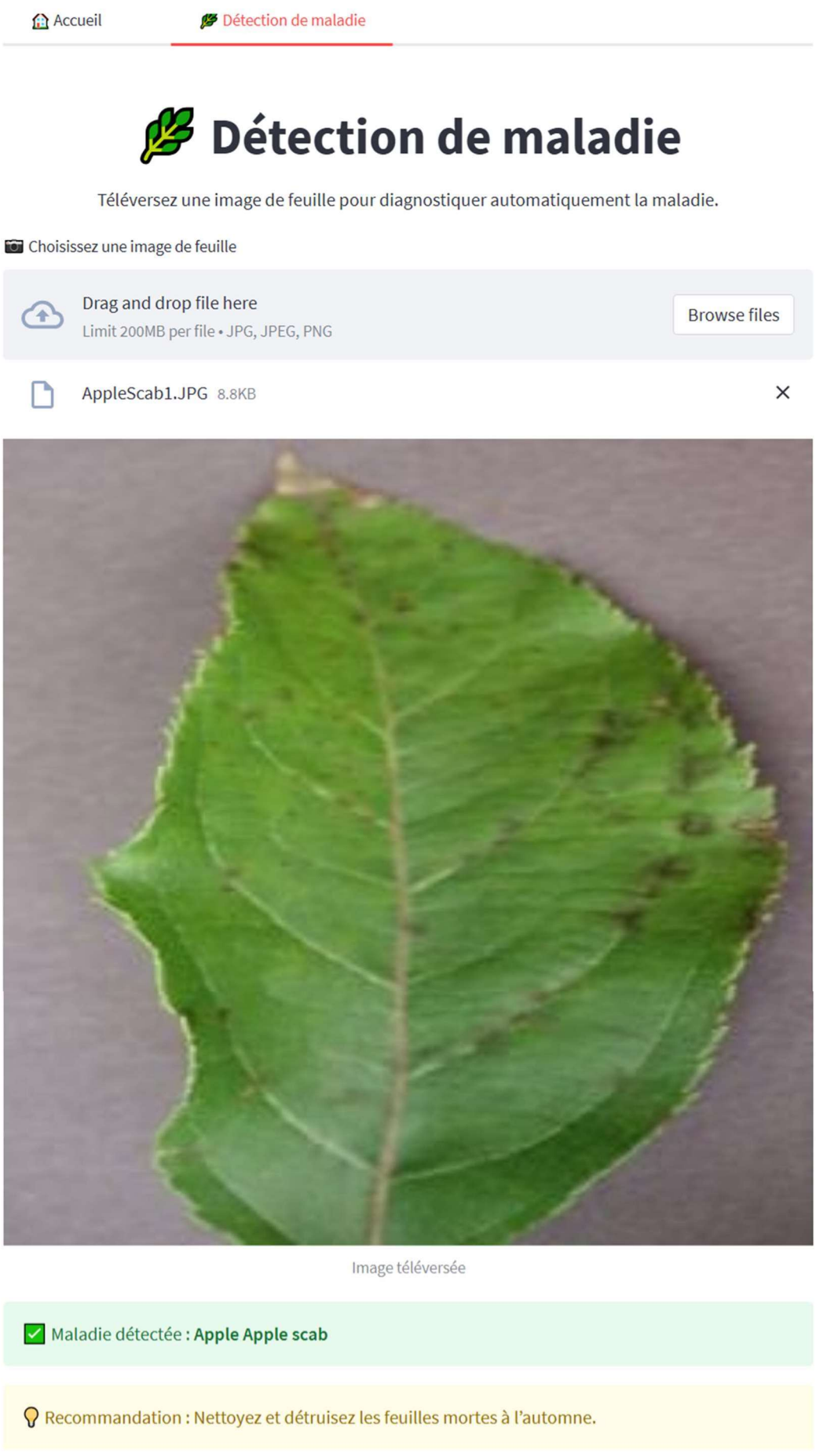

Figure 3.12 : Exemple de résultat obtenu sur l'interface

Ces figures démontrent le fonctionnement complet de notre application web. Le système réussit à transformer une simple image en un diagnostic clair et une recommandation exploitable, validant ainsi notre objectif de créer un outil d'aide au diagnostic simple et accessible.

3.7. Conclusion

Au terme de ce troisième chapitre consacré à l'implémentation et à l'évaluation, nous avons parcouru toutes les étapes pratiques de la réalisation de notre projet. En partant de la configuration de l'environnement de développement, nous avons préparé et chargé le jeu de données, défini les hyperparamètres, puis lancé l'entraînement de notre modèle CNN sur 10 époques.

L'évaluation des performances a révélé des résultats très concluants. L'analyse des courbes d'apprentissage a attesté d'une convergence stable et rapide, sans surapprentissage majeur. L'analyse détaillée par classe a, quant à elle, confirmé une très haute précision sur la grande majorité des 38 classes tout en nous permettant d'identifier les quelques cas de figure les plus complexes pour le modèle. Enfin la démonstration pratique a illustré avec succès comment le diagnostic du modèle peut être couplé à une recommandation concrète, validant ainsi l'utilité de notre approche.

Conclusion générale

Au terme de ce mémoire, nous avons exploré la conception et l'implémentation d'un système intelligent pour la détection des maladies végétales, en réponse à un enjeu agricole et économique majeur. L'objectif principal était de développer un outil d'aide au diagnostic, rapide et précis, capable de surmonter les limites des méthodes traditionnelles d'observation visuelle, souvent lentes et subjectives.

Notre démarche a débuté par une étude approfondie des fondements théoriques, allant des pathologies végétales à l'intelligence artificielle, pour aboutir au choix des réseaux de neurones convolutifs (CNN) comme technologie centrale de notre solution. Ce choix a été motivé par la performance avérée des CNN dans les tâches d'analyse et de classification d'images.

Le cœur de notre travail a consisté à concevoir et développer une architecture CNN sur mesure, entraînée à partir de zéro (from scratch). Ce modèle a été spécifiquement optimisé pour identifier 38 classes distinctes de maladies à partir d'un vaste jeu de données d'images foliaires. La phase d'implémentation, menée dans un environnement Python et exploitant des librairies comme TensorFlow et Keras, a permis de traduire cette conception en un modèle fonctionnel.

L'évaluation des performances a montré des résultats très prometteurs, le modèle atteignant une précision globale de 98 %. L'analyse a confirmé une convergence stable et la robustesse de l'architecture, tout en validant la capacité du système à fournir un diagnostic automatisé fiable à partir d'une simple image. Cette analyse a également permis d'identifier des axes d'amélioration pour les maladies aux symptômes similaires.

Ce projet illustre avec succès l'apport de l'apprentissage profond à l'agriculture en proposant un système de diagnostic rapide et fiable pour optimiser la gestion des cultures. Les perspectives d'évolution visent à renforcer sa performance, notamment par l'enrichissement continu du jeu de données pour améliorer sa robustesse, ainsi que par l'intégration de recommandations plus dynamiques pour en faire un véritable assistant agronomique.

Toutefois, la perspective la plus importante concerne le déploiement d'une application mobile native. Le développement d'une telle application permettrait de mettre cet outil directement entre les mains des agriculteurs sur le terrain. Ils pourraient ainsi utiliser l'appareil photo de leur smartphone pour obtenir un diagnostic instantané, même sans connexion internet, transformant ce prototype en une solution pratique et largement accessible.

Bibliographie

[1] University of Kentucky, College of Agriculture. “Agricultural Plant Diseases”. Consulté le 29 avril 2025, à : https://www.uky.edu/Ag/Entomology/PSEP/pdfs/11pests1disease.pdf

[2] Tjosvold, S. A. (2018). “The Disease Triangle: Fundamental Concept for Disease Management ”. UC Agriculture and Natural Resources” . Consulté le 28 avril 2025, à : https://ucanr.edu/blog/nursery-and-flower-grower/article/disease-triangle-fundamental-concept-disease-management#:~:text=Although%20these%20three%20plant%20hosts,rotated%20into%20the%20infested%20area.

[3] Neveln, V. (s.d.). “How to Spot 5 Common Plant Diseases and Keep Your Garden Healthy”. Better Homes&Gardens. Consulté le 29 avril 2025, à : https://www.bhg.com/gardening/pests/insects-diseases-weeds/common-plant-diseases/

[4] univ biskra. “Généralités sur le traitement d’images”. Consulté le 2 mai 2025, à : http://thesis.univ-biskra.dz/2271/6/Chapitre%2003.pdf.

[5] Khenfer,I et Bettayeb, E. et Salhi, D. (2020). Filtrage Adaptatif 2-D pour la Restauration d’images perturbées par du Bruit Impulsionnel. Mémoire de master. Université Kasdi Merbah Ouargla. Faculté des Nouvelles Technologies de l’Information et de la Communication, Département d’Electronique et de Communication.

[6] Pixees. (s.d). “ Photo et image – SNT”. Consulté le 2 mai 2025, à : https://pixees.fr/informatiquelycee/n_site/snt_photo_image.html

[7] Glover, E. (s.d.). “What Is Artificial Intelligence (AI)? ”. Built In. Consulté le 3 mai 2025, à :

https://builtin.com/artificial-intelligence

[8] Filimowicz, M. (2023). “10 Historical Milestones in the Development of AI Systems”. Medium.

Consulté le 3 mai 2025, à : https://medium.com/higher-neurons/10-historical-milestones-in-the-development-of-ai-systems-b99f21a606a9

[9] Ibm. (2021). “What is machine learning?”. Consulté le 3 mai 2025, à : https://www.ibm.com/think/topics/machine-learning

[10] Uc Berkeley School of Information. “What Is Machine Learning (ML)?”. Consulté le 3 mai 2025, à : https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/

[11] Biba, J. (s.d). “4 Types of Machine Learning to Know”. Builtin. Consulté le 4 mai 2025, à :

https://builtin.com/machine-learning/types-of-machine-learning

[12] Yasar, K. et S Gillis, A. et K. Pratt, M. “What is unsupervised learning?”. Techtarget. Consulté le 4 mai 2025, à : https://www.techtarget.com/searchenterpriseai/definition/unsupervised-learning

[13] Taylor, J. (s.d). “Definition of Semi-supervised learning”. Maddevs. Consulté le 4 mai 2025, à :

https://maddevs.io/glossary/semi-supervised-learning/

[14] Gavrilova, Y. (2024). “Reinforcement Learning: How It Works”. Consulté le 4 mai 2025, à : https://serokell.io/blog/reinforcement-learning-guide#defining-reinforcement-learning

[15] Shepard, J. (2023). “What are the four types of machine learning, and what are they used for?”. Microcontroller Tips. Consulté le 4 mai 2025, à : https://www.microcontrollertips.com/what-are-the-four-types-of-machine-learning-and-what-are-they-used-for/

[16] Grossi, E. et Buscema, M. (2008). Introduction to artificial neural networks. European Journal of Gastroenterology & Hepatology, vol 19, no 12, pp.1046-1054.

[17] Ibm. (2024). “What is deep learning?”. Consulté le 5 mai 2025, à : https://www.ibm.com/think/topics/deep-learning

[18] Robert, J. (2020). “Machine Learning vs Deep Learning : Quelles différences ? ”. Datascientest.

Consulté le 5 mai 2025, à : https://datascientest.com/quelle-difference-entre-le-machine-learning-et-deep-learning

[19] Ionos. (2025). “Recurrent Neural Network : fonctionnement et structure”. Consulté le 5 mai 2025 à : https://www.ionos.fr/digitalguide/sites-internet/developpement-web/recurrent-neural-network/

[20] Keita, Z. (2023). “An Introduction to Convolutional Neural Networks (CNNs)”. Consulté le 5 mai 2025 à : https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns

[21] Ultralytics. (s.d). “Réseau neuronal convolutif (CNN) ”. Consulté le 5 mai 2025 à : https://www.ultralytics.com/fr/glossary/convolutional-neural-network-cnn

[22] developersbreach. (2020). “Convolutional Neural Network | Deep Learning”. Consulté le 6 mai 2025 à : https://developersbreach.com/convolution-neural-network-deep-learning/

[23] Haque, K. (2023). “What is Convolutional Neural Network — CNN (Deep Learning)”. Consulté le 6 mai 2025 à : https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5

[24] Datascientest. (2023). “Convolutional Neural Network: Everything You Need to Know”. Consulté le 6 mai 2025 à : https://datascientest.com/en/convolutional-neural-network-everything-you-need-to-know

[25] Rosebrock, A. (2021). “Convolutional Neural Networks (CNNs) and Layer Types”. Consulté le 6 mai 2025 à : https://pyimagesearch.com/2021/05/14/convolutional-neural-networks-cnns-and-layer-types/

[26] Liakos, K. G., Busato, P., Moshou, D., Pearson, S., & Bochtis, D. (2018). Machine learning in agriculture: A review. Sensors, 18(8), 2674.

[27] Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. Computers and Electronics in Agriculture, 147, 70-90.

[28] Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., & Stefanovic, D. (2016). Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification. Conference paper, In 2016 24th telecommunications forum (TELFOR).

[29] Lu, J., & Tan, L., & Jiang, H. Review on Convolutional Neural Network (CNN) Applied to Plant Leaf Disease Classification. Agriculture 2021, 11, 707

[30] El Sakka, M., Ivanovici, M., Chaari, L., & Mothe, J. (2025). A Review of CNN Applications in Smart Agriculture Using Multimodal Data. *Sensors*, *25*(2), 472.

[31] Zebra. (s.d). “What Is the Difference Between Machine Learning and Deep Learning?”. Consulté le 13 mai 2025, à : https://www.zebra.com/us/en/resource-library/faq/what-is-the-difference-between-machine-learning-deep-Learning.html

[32] Codecademy. (s.d). “Rectified Linear Unit (ReLU) Function in Deep Learning”. Consulté le 13 mai 2025, à : https://www.codecademy.com/article/rectified-linear-unit-relu-function-in-deep-learning

[33] Belagatti, P. (2024). “Understanding the Softmax Activation Function: A Comprehensive Guide”. Consulté le 13 mai 2025, à : https://www.singlestore.com/blog/a-guide-to-softmax-activation-function/

[34] Hughes, C. (2024). “A Brief Overview of Cross Entropy Loss”. Medium. Consulté le 14 mai 2025, à : https://medium.com/@chris.p.hughes10/a-brief-overview-of-cross-entropy-loss-523aa56b75d5

[35] Helm, C. (2023). “Adaptive Moment Estimation : comprendre Adam et l'utiliser correctement ”. Konfuzio. Consulté le 14 mai 2025, à : https://konfuzio.com/fr/estimation-adaptative-du-moment/

[37] Aws. (s.d). “Qu'est-ce que Python ? ”. Consulté le 15 mai 2025, à : https://aws.amazon.com/fr/what-is/python/

[38] Databricks. (s.d). “TensorFlow ”. Consulté le 15 mai 2025, à : https://www.databricks.com/glossary/tensorflow-guide#:~:text=TensorFlow%20is%20an%20open%2Dsource,help%20it%20learn%20these%20tasks.

[39] Milvus. (s.d). “What is Keras, and how does it relate to TensorFlow? ”. Consulté le 15 mai 2025, à : https://milvus.io/ai-quick-reference/what-is-keras-and-how-does-it-relate-to-tensorflow

[40] Daniel. (2023). “NumPy : the most used Python library in Data Science”. Consulté le 15 mai 2025, à : https://datascientest.com/en/numpy-the-python-library-in-data-science

[41] GeeksforGeeks. (s.d). “What is python scikit library? ”. Consulté le 15 mai 2025, à : https://www.geeksforgeeks.org/machine-learning/what-is-python-scikit-library/

[42] Kumar, A. (2024). “MatplotLib In Python: Everything You Need to Know”. Consulté le 15 mai 2025, à : https://pwskills.com/blog/matplotlib-in-python/

[43] https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset/data

[44] Keras. (s.d). “Image data loading”. Consulté le 16 mai 2025, à : https://keras.io/api/data_loading/image/

[45] developers.google. (s.d). “Classification: justesse, rappel, précision et métriques associées ”. Consulté le 17 mai 2025, à : https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall?hl=fr#:~:text=Accuracy%20is%20the%20proportion%20of,out%20generic%20or%20unspecified%20tasks

[46] Shankar, R. (s.d). “Qu’est-ce qu’une matrice de confusion dans l’apprentissage automatique ? ”. Geekflare. Consulté le 18 mai 2025, à : https://geekflare.com/fr/confusion-matrix-in-machine-learning/

[47] Draelos, R. (2019). “Measuring Performance: The Confusion Matrix ”. Consulté le 18 mai 2025, à : https://glassboxmedicine.com/2019/02/17/measuring-performance-the-confusion-matrix/

[48] Jungseed. (s.d). “Cedar-Apple Rust - Solution Guide”. Consulté le 19 mai 2025, à : https://www.jungseed.com/category/cedar-apple-rust

[49] University of Minnesota Extension. (s.d). “Apple scab of apples and crabapples”. Consulté le 19 mai 2025, à : https://extension.umn.edu/plant-diseases/apple-scab#:~:text=Plant%20and%20prune%20correctly,trunk%20or%20within%20the%20canopy.

[50] Grow smart live. (2024). “A Guide to Corn Rusts: Southern Corn Rust and Common Corn Rust”. Consulté le 19 mai 2025, à : https://growsmartlive.com/news/5908

[51] Cropscience. (2024). “Early Blight of Potato”. Consulté le 19 mai 2025, à : https://www.cropscience.bayer.us/articles/cp/early-blight-potatoes

[52] Selvey, T. (s.d). “How to Identify, Control and Prevent Blight on Your Tomatoes”. Gardentech. Consulté le 19 mai 2025, à : https://www.gardentech.com/blog/pest-id-and-prevention/fight-blight-on-your-tomatoes#:~:text=Mulch%20around%20the%20base%20of%20the%20plant,and%20keeps%20blight%20from%20causing%20further%20damage.

[53] Plantix. (s.d). “Tomato Yellow Leaf Curl Virus”. Consulté le 19 mai 2025, à : https://plantix.net/en/library/plant-diseases/200036/tomato-yellow-leaf-curl-virus/