République Algérienne Démocratique Et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université du 20 Août 55 –Skikda

Faculté des sciences

Département d'informatique

Mémoire de fin d'étude en vue de l'obtention du diplôme de Master en informatique

Option : Systèmes informatiques (SI)

Thème :

Détection automatisée des maladies de la pomme de terre par Deep Learning

Réalisé par :

- ✓ **Chebel Seif Eddine**

Encadré par :

M^me^ Magroun Hanane

Session : Juin 2025


بسم الله الرحمن الرحيم

Remerciements

Avant toute chose, que les louanges soient adressées à Allah, le Très-Haut, le Sage, qui nous a comblés de Ses bienfaits visibles et invisibles, et qui nous a accordé la force, la patience et la clarté d'esprit pour mener à bien ce travail. Sans Sa grâce et Son soutien, rien de cela n'aurait été possible.

Nous tenons ensuite à exprimer notre profonde gratitude à toutes les personnes qui, de près ou de loin, ont contribué à la réalisation de ce mémoire et ont soutenu notre parcours académique.

Nos remerciements les plus sincères vont à Madame **MAGROUN Hanane**, mon encadrante, pour sa confiance, ses conseils judicieux, sa bienveillance et sa disponibilité constante. Son accompagnement a été d'un grand appui tout au long de ce travail.

Nous remercions également les membres du jury pour l'intérêt qu'ils ont porté à notre mémoire et pour l'honneur qu'ils nous font en acceptant de l'évaluer.

Enfin, nous adressons notre reconnaissance à l'ensemble du corps enseignant pour leur engagement, la qualité de leur enseignement et leur précieuse contribution à notre formation

Dédicace

*Je dédie ce modeste travail à mes chers parents, À **mon père**, pour ses paroles pleines de sagesse, son soutien constant et la fierté qu'il m'inspire chaque jour. À **ma mère**, pour son amour infini, sa foi inébranlable en moi, et pour m'avoir appris que rien n'est impossible.*

« Merci pour tout ce que vous m'avez donné »

*À ma chère sœur **Mouna**, Je tiens à t'adresser mes plus sincères remerciements pour ton aide précieuse dans la réalisation de ce mémoire, et pour ton soutien indéfectible tout au long de mon parcours universitaire. Tu as toujours été là pour m'encourager, me motiver et croire en moi, même dans les moments les plus difficiles. Ce travail est aussi le fruit de ta bienveillance et de ton amour constant. « Merci du fond du cœur »*

*À mon petit frère **Abd Raouf**, dont la présence m'a toujours réconforté et motivé.*

*À mes amis très chers, **Hamza** et **Aymen**, qui ont été pour moi comme des frères, toujours présents avec leur gentillesse, leur soutien et leurs encouragements.*

*À mon professeur estimé, **M. LAMRED**,Je vous adresse ma profonde gratitude pour l'impact significatif que vous avez eu sur ma vie personnelle et mon parcours académique.*

Votre soutien constant, vos encouragements sincères et vos conseils avisés ont été pour moi une source d'inspiration et de motivation. Merci pour votre bienveillance, votre dévouement et l'humanité que vous incarnez.

*À mon oncle bien-aimé **Kamel**, qui a toujours été pour moi un second père, un véritable pilier tout au long de ma vie. Tes paroles ont un écho particulier dans mon cœur, et tes conseils ont laissé une empreinte profonde en moi. Merci du fond du cœur pour ton soutien inconditionnel, ta présence rassurante et ton amour sincère.*

Un remerciement spécial à toute ma famille et à mes amis pour leur amour, leur soutien et leurs encouragements constants tout au long de mon parcours.

الملخص

تُعد مشكلة اكتشاف أمراض النباتات من التحديات الجوهرية في مجال الزراعة الذكية، لما لها من أثر مباشر على الإنتاجية وتقليل الخسائر الاقتصادية. ومع التقدم الكبير في تقنيات الذكاء الاصطناعي، أصبح من الممكن توظيف خوارزميات التعلم الآلي، لا سيما التعلم العميق، في تحليل صور النباتات وتشخيص الأمراض بدقة عالية. تهدف هذه المذكرة إلى تصميم وتطوير نظام ذكي يعتمد على نماذج الشبكات العصبية التلافيفية (CNN)لتصنيف أمراض أوراق البطاطا بدقة وفعالية .تم اعتماد نموذجين أساسيين في هذا العمل، هما :DenseNet121وU-Net، وذلك لتقييم أدائهما في مهام تصنيف واكتشاف الأمراض. شملت مراحل العمل معالجة وتحضير بيانات مجموعة Plant Village ، التي تحتوي على صور مصنفة لأوراق بطاطا سليمة ومصابة، بالإضافة إلى خطوات المعالجة المسبقة كإزالة الضجيج، وتوحيد الأبعاد، وتسوية الإضاءة، قبل المرور إلى استخراج السمات وبناء النماذج، كما تم دمج النموذج النهائي مع واجهة ويب تفاعلية تم تطويرها باستخدام تقنيات حديثة مثل Fast API وReactJS، لتوفير منصة سهلة الاستخدام تتيح للمستخدم التفاعل مع النظام بشكل مباشر. أظهرت النتائج أن نموذج DenseNet121حقق أعلى أداء بدقة بلغت 99.48%أثناء التدريب، مع استقرار جيد في نتائج التحقق، ما يعكس فعاليته العالية في التصنيف. في المقابل، أظهر نموذج U-Netبعض التذبذب في الأداء مما يجعله أقل استقرارًا .تؤكد هذه الدراسة فعالية استخدام تقنيات الذكاء الاصطناعي في تطوير حلول تطبيقية موجهة للمجال الزراعي، وتمهد الطريق نحو أدوات ذكية لمساعدة المزارعين في المستقبل.

الكلمات المفتاحية :التعلم العميق، أمراض النباتات، تصنيف الصور، DenseNet121، U-Net، الذكاء الاصطناعي، Plant Village، معالجة الصور.

Abstract

Plant disease detection is considered a vital challenge in the domain of smart agriculture due to its significant impact on productivity and economic sustainability. With the rapid advancements in artificial intelligence, particularly in deep learning, it has become feasible to automatically analyze plant images and accurately identify diseases. This thesis focuses on the design and implementation of an intelligent system for detecting and classifying potato leaf diseases using Convolutional Neural Network (CNN)-based deep learning models, specifically DenseNet121 and U-Net. The methodology began with the preparation and preprocessing of the Plant Village dataset, which includes annotated images of both healthy and diseased leaves. Essential preprocessing steps such as denoising, normalization, and resizing were applied prior to feature extraction and model training. To ensure usability, the proposed system was integrated with a modern web interface developed using FastAPI and ReactJS, allowing end users to interact with the classification model in real time. Experimental results demonstrated that the DenseNet121 model outperformed U-Net, achieving a high training accuracy of 99.48% with stable validation performance, thus proving its robustness in disease classification. Conversely, the U-Net model exhibited some fluctuations, indicating less consistency. This work highlights the potential of deep learning in providing practical, AI-powered solutions for the agricultural sector and contributes toward the development of smart tools for future farmers.

Keywords: Deep Learning, Plant Disease Detection, DenseNet121, U-Net, CNN, Artificial Intelligence, Image Processing, Plant Village.

Résumé

La détection des maladies des plantes constitue un enjeu majeur dans le domaine de l'agriculture intelligente, car un diagnostic précoce peut améliorer considérablement la productivité et réduire les pertes économiques. Avec l'évolution rapide des technologies d'intelligence artificielle, il est désormais possible d'analyser automatiquement des images de plantes et d'identifier les maladies avec une grande précision. Ce mémoire porte sur la conception et le développement d'un système intelligent permettant la détection et la classification des maladies des feuilles de pomme de terre, en s'appuyant sur des modèles d'apprentissage profond basés sur les réseaux de neurones convolutifs (CNN), notamment DenseNet121 et U-Net. La méthodologie adoptée commence par le traitement et la préparation de la base de données Plant Village, qui contient des images étiquetées de feuilles saines et malades. Les étapes de prétraitement incluent la réduction du bruit, la normalisation et le redimensionnement des images, suivies par l'extraction des caractéristiques et l'entraînement des modèles. Pour faciliter l'utilisation du système une interface web interactive a été développée à l'aide de technologies modernes telles que FastAPI et ReactJS, offrant ainsi une expérience conviviale pour l'utilisateur.. Les résultats expérimentaux ont montré que le modèle DenseNet121 a obtenu les meilleures performances avec une précision d'entraînement atteignant 99,48 %, et une stabilité remarquable lors de la validation, ce qui démontre son efficacité dans la classification des maladies. En revanche, le modèle U-Net a présenté des performances moins stables, avec certaines fluctuations. Ce travail confirme le potentiel de l'intelligence artificielle dans le développement de solutions pratiques destinées au secteur agricole, et ouvre la voie à des outils intelligents au service des agriculteurs de demain.

Mots-clés : Apprentissage profond, Détection des maladies des plantes, DenseNet121, U-Net, CNN, Traitement d'images, Intelligence artificielle, Plant Village.

Sommaire

Liste des figures

Chapitre 01

Chapitre 02

Chapitre 03

Liste des Tableaux

Chapitre 3

Liste des Abréviations

IA : Intelligence Artificielle

ML: Machine Learning

DL: Deep Learning

CNN: Convolutional Neural Network

RNN: Recurrent Neural Network

ANN: Artificial Neural Network

SVM: Support Vector Machine

KNN: K-Nearest Neighbors

API: Application Programming Interface

GPU: Graphics Processing Unit

CPU: Central Processing Unit

HTML: Hypertext Markup Language

CSS: Cascading Style Sheets

JS: JavaScript

IDE: Integrated Development Environment

HTTP: Hypertext Transfer Protocol

JSON : JavaScript Object Notation

UI : User Interface

UX : User Expérience

TF/Keras : TensorFlow / Keras

FastAPI : Framework Python pour la création d'API

ReactJS : Bibliothèque JavaScript pour les interfaces utilisateur

Node.js : Environnement d'exécution JavaScript côté serveur

PlantVillage : Base de données d'images de feuilles saines et malades

EB: Early Blight (Brûlure précoce)

LB: Late Blight (Brûlure tardive)

H: Healthy Plants (Plantes saines)

DSA : Data Augmentation (Augmentation des données)

Introduction générale

Introduction générale

L'intelligence artificielle (IA) et l'apprentissage automatique (machine learning) ont connu un essor considérable ces dernières années, avec des applications croissantes dans divers domaines, notamment la vision par ordinateur et la reconnaissance d'images. L'un des cas d'usage les plus prometteurs concerne la détection automatique des maladies des plantes à partir d'images de leurs feuilles. Ce domaine, à l'intersection entre l'agriculture de précision et les technologies intelligentes, représente une solution efficace aux pertes agricoles causées par des maladies non détectées à temps. Les maladies des plantes constituent une menace sérieuse pour la sécurité alimentaire mondiale. Les méthodes traditionnelles de diagnostic reposent souvent sur l'expertise humaine, coûteuse et sujette à l'erreur, ou sur des tests de laboratoire peu accessibles dans les zones rurales. L'apprentissage profond (Deep Learning), et plus précisément les réseaux de neurones convolutifs (CNN), offrent une alternative performante pour automatiser l'identification des maladies à partir de simples images.

Problématique

Malgré les avancées technologiques, plusieurs défis subsistent dans la reconnaissance des maladies des plantes : la ressemblance entre certains symptômes, les variations d'apparence selon l'éclairage, l'angle de la photo ou encore la qualité de l'image. Cela soulève la question suivante :

Comment concevoir un système intelligent, basé sur les CNN, capable d'identifier avec précision les maladies des feuilles de plantes, malgré la diversité des données et les conditions de prise de vue variées ?

Objectifs

Ce travail vise à :

- Étudier les concepts fondamentaux de l'intelligence artificielle, de l'apprentissage automatique et de l'apprentissage profond.
- Explorer l'utilisation des réseaux de neurones convolutifs (CNN) pour la reconnaissance d'images.

- Implémenter un système de classification capable de détecter les maladies des plantes à partir d'images de feuilles, en utilisant des modèles CNN tels que DenseNet et U-Net.
- Développer une interface web simple permettant aux utilisateurs de diagnostiquer automatiquement les maladies à partir de leurs propres images.

Plan du mémoire

Ce mémoire est structuré en trois chapitres principaux :

- **Chapitre 1 :** Présente un aperçu général des techniques d'apprentissage automatique, de l'intelligence artificielle à l'apprentissage profond, en passant par les différents types d'algorithmes.
- **Chapitre 2 :** Traite spécifiquement de la détection des maladies des plantes, en détaillant les étapes de traitement d'image et en passant en revue les travaux existants.
- **Chapitre 3 :** Décrit notre solution proposée, depuis la collecte des données jusqu'à la mise en œuvre du modèle CNN, l'analyse des performances et le développement de l'interface utilisateur.
- L'annexe présente Environnement de travail et outils de développement

Chapitre 1 :

Aperçu Des Techniques D'apprentissage Automatique

1.1 Introduction

L'intelligence artificielle et l'apprentissage automatique ont transformé de nombreux domaines en utilisant les données de manière plus efficace. La capacité de l'IA à acquérir des connaissances, à penser de manière créative et à résoudre des problèmes a considérablement amélioré l'automatisation des industries, leurs capacités de résolution de problèmes et leur efficacité. Dans ce chapitre, nous aborderons les fondamentaux de l'IA et de l'apprentissage automatique en nous appuyant sur des méthodes traditionnelles telles que l'apprentissage supervisé, non supervisé et par renforcement. Nous examinerons également des techniques avancées comme les cadres d'apprentissage profond, y compris les réseaux de neurones artificiels, les réseaux de neurones convolutionnels et les réseaux de neurones récurrents. De plus, nous analyserons comment les algorithmes jouent un rôle essentiel dans les décisions intelligentes prises par les systèmes d'IA. Les algorithmes d'apprentissage profond constituent la base des technologies d'IA qui influencent une large gamme d'industries, notamment la santé, le divertissement et la finance.

1.2 Intelligence artificielle

1.2.1 Définition de l'intelligence

L'intelligence fait référence à la capacité d'un organisme à apprendre, donner du sens, résoudre des problèmes, etc. Elle comprend différentes capacités telles que la réflexion, la mémorisation, la créativité et le choix. Un autre aspect important dans la définition de l'intelligence concerne l'intelligence émotionnelle d'une personne et sa capacité à établir des connexions sociales, etc.

De nombreux psychologues débattent de la question de savoir si l'intelligence est une capacité générale ou une combinaison d'intelligences. La théorie des intelligences multiples a été proposée par Howard Gardner (1983). Il a affirmé que l'intelligence existe sous de nombreuses formes. Premièrement, l'intelligence logico-mathématique.

Deuxièmement, l'intelligence linguistique. Troisièmement, l'intelligence spatiale. En outre, il existe des intelligences musicales, kinesthésiques, interpersonnelles et intrapersonnelles. Les chercheurs continuent de débattre pour savoir si l'intelligence est une capacité cognitive appliquée à différents domaines [1].

1.2.2 Définition de l'intelligence artificielle (IA)

L'intelligence artificielle (IA) désigne les machines ou systèmes informatiques capables d'exécuter des tâches habituellement réalisées par des humains, telles que l'apprentissage, le raisonnement, la résolution de problèmes et la prise de décisions [2]. John McCarthy a inventé le terme en 1955 et a déclaré que l'IA est « la science et l'ingénierie de la création de machines intelligentes ».

De nombreuses personnes débattent pour savoir si l'IA est réellement intelligente ou si elle est simplement une suite de calculs complexes. Comme nous avons défini l'intelligence comme la capacité à résoudre des problèmes, une partie des personnes estime que les IA devraient être considérées comme intelligentes [3]. Néanmoins, d'autres soutiennent que les IA sont fondamentalement différentes des humains et ne possèdent ni conscience, ni émotions, ni créativité [4].

Par exemple, une simple calculatrice de poche effectue des calculs mathématiques. De même, un système IA effectue également des calculs complexes tels que la résolution de problèmes et l'apprentissage automatique. Pourtant, nous ne qualifions pas la calculatrice de « intelligente ».

Ainsi, nous ne considérons quelque chose comme intelligent que s'il apprend. C'est la grande différence entre l'IA et une calculatrice. Les systèmes d'IA sont capables d'améliorer leurs performances au fil du temps.

1.2.3 IA et Algorithmes

Les algorithmes sont des méthodes pas à pas pour résoudre des problèmes et traiter des données. Ils aident l'IA à tirer des conclusions et à répondre à des questions. Les algorithmes peuvent être très simples, comme ceux utilisés dans les calculs de base, ou très complexes, permettant à l'IA de traiter de grandes quantités d'informations et de prendre des décisions de manière autonome [5].

Les algorithmes qui apprennent de l'expérience sont utilisés dans l'IA. Votre aspirateur robot, qui apprend continuellement à éviter les obstacles, en est un parfait exemple. Avec l'avancée de la technologie de l'IA, il est prévu d'augmenter l'efficacité dans divers domaines en permettant aux humains de résoudre des problèmes complexes en moins de temps et avec plus de précision [6].

1.3 Apprentissage automatique

L'apprentissage automatique (ML), en tant que sous-ensemble de l'intelligence artificielle, tire parti de la capacité à apprendre à partir des données, également appelée reconnaissance de motifs. Contrairement aux systèmes traditionnels, les modèles ML ne sont pas limités par des règles fixes ou des hypothèses ils apprennent plutôt à partir d'exemples et s'auto-corrigent de manière axée sur les données.

L'apprentissage automatique permet une conception efficace, fiable et rentable [7]. Sa capacité à générer rapidement des modèles est un atout considérable [8]. Il peut produire des modèles à partir de données rapidement et précisément. En outre, l'apprentissage automatique peut rapidement, précisément et efficacement extraire des motifs à partir des données elles-mêmes. Cela permet le traitement de grandes quantités de données, telles que les données de santé [9].

Les données ML proviennent de multiples sources telles que des serveurs réseau, des dossiers médicaux électroniques (DME), des bases de données génomiques, des PC, des applications mobiles, des smartphones, des capteurs et des dispositifs portables [10].

1.3.1 Types de techniques d'apprentissage automatique

1.3.1.1 Apprentissage supervisé

L'apprentissage supervisé implique une approche d'apprentissage automatique lorsque les données sont présentées sous forme d'entrées (caractéristiques) et de sorties (résultats). L'algorithme supervisé apprend la fonction de mappage qui associe l'entrée à la sortie, utilisée pour les prévisions.

Pour être efficace, l'apprentissage supervisé nécessite de grands ensembles de données avec des exemples étiquetés. Dans l'apprentissage supervisé, chaque point de données dans l'ensemble de données contient l'entrée ainsi que la sortie correcte. Cela aide le modèle à apprendre par l'exemple. Cependant, obtenir et étiqueter de grands ensembles de données peut être coûteux et chronophage, surtout dans les domaines où il est difficile de trouver des données non étiquetées et où leur obtention ou leur classification est onéreuse.

L'apprentissage supervisé peut être catégorisé en deux types principaux :

a. Classification

Dans les tâches de classification, la variable de sortie appartient à un ensemble de catégories prédéfinies. L'objectif du modèle est d'assigner de nouvelles entrées à l'une de ces

catégories. Par exemple, un modèle IA peut distinguer entre des images de chats et de chiens ou classifier des sentiments textuels comme positifs ou négatifs.

b. Régression

Les tâches de régression impliquent la prédiction d'une valeur numérique continue plutôt que d'une catégorie discrète. Par exemple, la prédiction du prix d'une maison en fonction de facteurs tels que l'emplacement et la taille, la prévision des tendances boursières ou l'estimation de la demande future d'un produit [11].

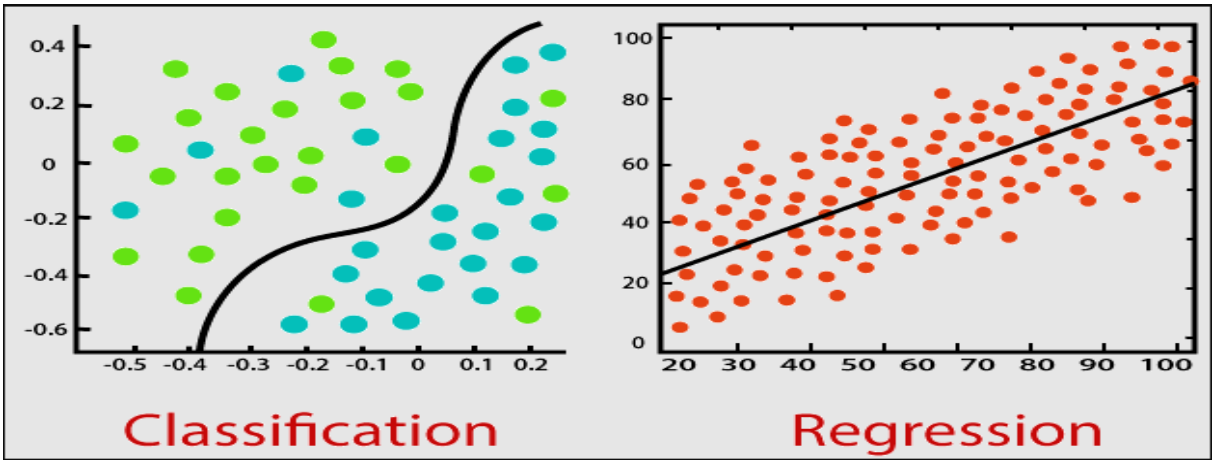


Figure 1.1 : Illustration graphique de la régression et de la classification [12]

1.3.1.2 Apprentissage non supervisé

Contrairement à l'apprentissage supervisé, l'apprentissage non supervisé est utilisé lorsqu'il n'y a pas de données de sortie étiquetées. Au lieu de cela, le modèle analyse les motifs et les relations dans les données afin de découvrir leur structure sous-jacente. L'objectif de l'apprentissage non supervisé est d'identifier des motifs cachés sans intervention humaine.

L'une des techniques clés de l'apprentissage non supervisé est le clustering, où des points de données similaires sont regroupés en fonction de caractéristiques communes. Par exemple, le clustering peut être utilisé pour segmenter les clients en différents groupes en fonction de leur comportement d'achat, aidant ainsi les entreprises à optimiser leurs stratégies marketing.

Un autre exemple : dans le domaine médical, l'apprentissage non supervisé peut aider à identifier de nouveaux motifs de maladies basés sur des symptômes communs, même s'ils n'ont pas été explicitement classifiés auparavant [13].

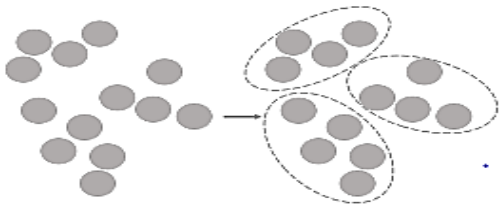


Figure 1.2 : Schéma de l'apprentissage non supervisé [13].

1.3.1.3 Apprentissage semi-supervisé

Comme son nom l'indique, l'apprentissage semi-supervisé est une approche intermédiaire entre l'apprentissage supervisé et non supervisé. Il combine des données étiquetées et non étiquetées dans le processus d'entraînement. En général, une petite quantité de données étiquetées est disponible, ainsi qu'un grand volume de données non étiquetées.

Une procédure courante consiste à regrouper d'abord des points de données similaires à l'aide d'un algorithme d'apprentissage non supervisé, puis à utiliser le petit ensemble de données étiquetées pour attribuer des étiquettes aux données non étiquetées restantes.

Exemple pratique : dans la reconnaissance d'écriture manuscrite, seuls quelques mots étiquetés manuellement peuvent être disponibles, tandis que le reste du jeu de données reste non étiqueté. Un algorithme d'apprentissage semi-supervisé peut utiliser les données étiquetées pour inférer automatiquement les étiquettes du reste du jeu de données.

Cette approche est particulièrement utile dans les cas où l'obtention de données étiquetées est coûteuse ou chronophage, comme la classification des images médicales ou l'analyse des données génétiques [14].

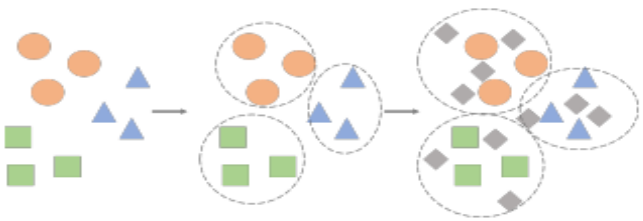


Figure 1.3 : Schéma de l'apprentissage semi-supervisé [14]

1.3.1.4 Apprentissage par renforcement

L'apprentissage par renforcement (RL) diffère des autres techniques d'apprentissage en ce qu'il est basé sur une approche « essai-erreur » pour atteindre un objectif. Dans le RL, un agent artificiel interagit avec un environnement, prend des décisions séquentielles et reçoit des récompenses ou des pénalités en fonction de ses actions. L'objectif est de maximiser les récompenses cumulatives au fil du temps.

L'apprentissage par renforcement est largement utilisé dans diverses applications, notamment :

• **Jeux vidéo** : Les agents IA peuvent apprendre à jouer à des jeux vidéo et développer des stratégies basées sur des expériences passées. Par exemple, Alpha GO, développé par DeepMind, a battu des champions du monde au jeu de Go en utilisant l'apprentissage par renforcement.

• **Robotique** : Le RL est utilisé pour entraîner des robots à effectuer des tâches telles que naviguer dans des environnements inconnus, saisir des objets et assister dans des entrepôts.

• **Véhicules autonomes** : Les voitures autonomes s'appuient sur l'apprentissage par renforcement pour prendre des décisions en temps réel, telles que l'évitement d'obstacles et la réponse aux signaux de circulation.

L'un des principaux avantages de l'apprentissage par renforcement est qu'il ne nécessite pas de données d'entraînement pré-étiquetées. Au lieu de cela, l'IA apprend à partir des interactions avec son environnement. Cependant, le RL peut être coûteux en termes de calcul, car les modèles d'IA nécessitent de nombreuses itérations pour atteindre des performances optimales [15].

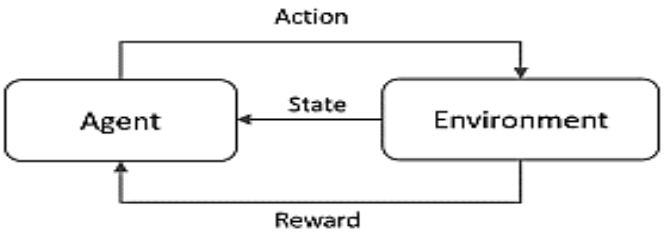


Figure 1.4 : Schéma de l'apprentissage par renforcement [15]

1.3.2 Les Modèles d'apprentissage automatique

Les modèles d'apprentissage automatique englobent une gamme de techniques conçues pour analyser des données, identifier des motifs et faire des prédictions basées sur les données d'entrée. Ces modèles sont catégorisés en différents types, chacun ayant une approche d'apprentissage et une méthodologie de traitement des données distinctes, notamment :

1.3.2.1 Les Modèles d'apprentissage supervisé

Les modèles d'apprentissage supervisé comprennent diverses techniques conçues pour analyser des données étiquetées et apprendre les relations entre les entrées et les sorties. Parmi les modèles les plus connus, on trouve :

• Vecteur à support machine (SVM)

Le Support Vector Machine (SVM) est un algorithme d'apprentissage supervisé utilisé pour les tâches de classification. Il fonctionne en trouvant l'hyperplan optimal qui sépare les points de données en deux catégories avec la plus grande marge possible. Le SVM est réputé pour sa grande précision et ses garanties théoriques contre le surapprentissage. Il gère efficacement les espaces de haute dimension en utilisant des fonctions noyau pour transformer les données dans un format plus facilement séparable.

Le SVM est un classificateur linéaire, ce qui signifie que les données (telles que les documents texte) doivent idéalement être linéairement séparables. Chaque document est représenté comme un point dans un espace vectoriel, et l'objectif est de trouver le meilleur séparateur (une ligne, un plan ou un hyperplan) qui divise les données.

La marge, qui est la distance entre le séparateur et les points de données les plus proches (vecteurs de support), doit être maximisée pour améliorer la généralisation. Bien que le SVM donne de bons résultats sur de nouvelles données, son temps d'entraînement peut être élevé [16].

- **Concepts clés**

• **Hyperplan :** Une frontière qui sépare différentes classes. L'hyperplan optimal est celui qui maximise la marge.

• **Vecteurs de support :** Les points de données les plus proches de l'hyperplan, qui influencent sa position et son orientation.

Le SVM simplifie la classification en mappant les données dans un espace de dimension supérieure où une séparation claire peut être trouvée. Il est largement utilisé en raison de son efficacité et de sa précision.

- **Avantages du SVM**

  • Solide fondement théorique.

  • Efficace dans les espaces de haute dimension [17].

  • Supporte différentes fonctions noyau pour la classification non linéaire [17].

- **Limites du SVM**

  • Nécessite des calculs mathématiques complexes [17].

  • Coût computationnel élevé lors de l'entraînement et du test.

  • Bayes naïf (NB)

Le Bayes naïf (NB) est une approche probabiliste basée sur le théorème de Bayes, supposant que les caractéristiques sont indépendantes (d'où le terme « naïf »). Il prédit les résultats futurs en se basant sur des données passées et des connaissances préalables.

Le NB est largement utilisé dans la classification de texte en raison de son efficacité computationnelle, de ses performances prédictives et de sa simplicité. Cependant, il est très sensible à la sélection des caractéristiques (termes utilisés pour la classification).

- **Comment fonctionne le Bayes naïf**

Le Bayes naïf est un classificateur probabiliste qui repose sur le principe de Bayes et l'hypothèse d'indépendance des caractéristiques. Il est couramment implémenté dans des langages de programmation comme Java en raison de sa simplicité, nécessitant uniquement des calculs de probabilité de base.

- **Avantages du NB**
  - Simple et facile à implémenter.
  - Plus rapide que de nombreux autres modèles de classification, surtout lorsque l'hypothèse d'indépendance est respectée.
  - Même si l'hypothèse d'indépendance n'est pas entièrement vraie, le NB fonctionne bien en pratique, maintenant une précision de classification stable.

Théorème de Bayes dans la classification NB :

Pour calculer la probabilité qu'une instance appartienne à une classe particulière, la règle de Bayes est appliquée comme suit [18] :

$$\mathbf{P(C_i|x) = \frac{P(x|C_i)P(C_i)}{P(x)}} \qquad \mathbf{(1.1)}$$

Où :

• P(Ci) : Probabilité a priori de la classe Ci.

• P(x) : Probabilité d'observer le vecteur de caractéristiques x.

• P(x|Ci) : Probabilité de x étant donné la classe Ci.

Le vecteur de caractéristiques x est attribué à la classe Ci si [16] :

$$\mathbf{\forall j \neq i, P(C_i|x) > P''(C_i|x)} \qquad \mathbf{(1.2)}$$

Relation entre l'analyse discriminante et le Bayes naïf:

L'analyse discriminante est un cas particulier de l'approche bayésienne, où les données d'entraînement sont modélisées à l'aide d'une distribution gaussienne. Une fois les paramètres estimés, des fonctions discriminantes sont construites pour classifier tout nouveau vecteur de caractéristiques [18] :

• Régression linéaire

La régression linéaire est une méthode statistique utilisée pour établir une relation linéaire entre la variable indépendante X et la variable dépendante Y. L'objectif est de trouver une fonction linéaire optimale, c'est-à-dire de déterminer un ensemble de coefficients (poids) afin que la fonction puisse prédire la valeur de la variable dépendante aussi précisément que possible. Formellement, le modèle de régression linéaire peut être exprimé comme suit :

$$\mathbf{Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots\ldots + \beta_n X_n + \varepsilon} \qquad \mathbf{(1.3)}$$

où β_0 est l'interception, β_1 à βn sont les coefficients de régression, X_1 à X_n sont les variables indépendantes, et ε est le terme d'erreur [19].

• Arbres de décision (DT)

Un arbre de décision (DT) est un algorithme d'apprentissage automatique qui représente les données sous forme d'une structure arborescente.

Un arbre est un graphe acyclique, connecté et non dirigé, composé d'un ensemble de nœuds catégorisés en trois types :

• Nœud racine : Le point de départ de l'arbre.

• Nœuds internes : Points de décision intermédiaires.

• Nœuds de feuille : Les derniers nœuds représentant les résultats de la classification.

Les arbres de décision apprennent à partir des observations (exemples), qui consistent en des attributs et des classes associées. Ils organisent les données dans une séquence hiérarchique de décisions, visant à identifier les similarités et les différences entre les valeurs des attributs au sein d'un ensemble de données [20].

Certains des algorithmes d'arbres de décision les plus utilisés sont :

• ID3 (Iterative Dichotomiser 3)

• C4.5 (Successeur de ID3)

• CART (Classification and Regression Trees) [21]

Étapes dans l'apprentissage des arbres de décision

1. Sélectionner une variable de segmentation
2. Traiter les variables continues
3. Définir la taille optimale de l'arbre
4. Prendre des décisions
5. Fusionner les nœuds lors de la segmentation

Les arbres de décision fournissent une approche structurée de la classification, ce qui les rend particulièrement utiles dans diverses applications, y compris la classification de texte, le diagnostic médical et l'analyse du comportement des clients.

• Réseaux de neurones

Un réseau de neurones est un modèle mathématique inspiré du cerveau humain. Il est constitué de neurones interconnectés qui aident à résoudre des problèmes complexes tels que la reconnaissance de formes et le traitement du langage naturel en ajustant les coefficients de

poids pendant la phase d'apprentissage. Bien qu'il imite la puissance de calcul du cerveau humain, il ne possède pas d'émotions [22].

Architecture de base des réseaux de neurones :

Un réseau de neurones se compose généralement de trois types de couches :

- Couche d'entrée : Reçoit les données brutes.
- Couches cachées : Traitent l'information à travers des connexions pondérées.
- Couche de sortie : Produit le résultat final.

Un réseau de neurones est caractérisé par :

- Le type de neurones utilisés.
- La règle d'apprentissage qui met à jour les poids.
- L'architecture, qui définit comment les neurones sont connectés.

Chaque neurone artificiel traite les valeurs d'entrée (z1, z2..., zn) en calculant une somme pondérée et en appliquant une fonction d'activation f pour déterminer la sortie O :

$$\boldsymbol{O} = (\sum_{i=1}^{n} \mathbf{w}_i \, \mathbf{z}_i) \; = \; \boldsymbol{f}(\boldsymbol{neti}) \qquad \textbf{(1.4)}$$

Où w_i sont les poids synaptiques attribués à chaque entrée z_i. La fonction d'activation est généralement une fonction sigmoïde.

Le processus d'apprentissage vise à ajuster ces poids en utilisant des exemples d'entraînement afin de minimiser une fonction d'erreur, qui représente la différence entre la sortie réelle du réseau et la sortie souhaitée [23].

- **K-Nearest Neighbors (KNN)**

K-Nearest Neighbors (KNN) est un algorithme de classification basé sur les exemples, où un document non vu est classé en fonction de la catégorie majoritaire de ses K documents d'entraînement les plus similaires. La similarité est mesurée à l'aide de mesures de distance telles que la distance euclidienne.

Mesures de distance utilisées dans KNN :

Soit Xp = (x_{p1}, x_{p2}..., x_{pn}) le vecteur de caractéristiques de l'entité p, et soient p et q deux entités comparées. KNN utilise couramment :

- Euclidean Distance:

$$\boldsymbol{D(Xp, Xq) = \sqrt{\sum_{i=1}^{N}(xp_i - xq_i)^2}} \qquad \textbf{(1.5)}$$

- Manhattan Distance:

$$\boldsymbol{D(Xp, Xq) = \sum_{i=1}^{N} |xp_i - xq_i|} \qquad \textbf{(1.6)}$$

- Minkowski Distance (generalized form):

$$\boldsymbol{D(Xp, Xq) = \left(\sum_{i=1}^{n} (x_{pi} - x_{qi})^r\right)^{1/r}} \qquad \textbf{(1.7)}$$

- Chebyshev Distance:

$$\boldsymbol{D(Xp, Xq) = \max|x_{pi} - x_{qi}|} \qquad \textbf{(1.8)}$$

Étapes de l'algorithme KNN :

1. Charger le jeu de données.
2. Définir la valeur de K.
3. Calculer la distance entre le point de test et tous les points d'entraînement.
4. Trier les points d'entraînement par ordre croissant de distance.
5. Sélectionner les K voisins les plus proches.
6. Identifier la classe la plus fréquente parmi ces voisins.
7. Attribuer le point de test à cette classe.
8. Fin de l'algorithme.

KNN peut également utiliser la similarité cosinus comme mesure de distance. Le choix de K est crucial : un K très petit peut entraîner un surapprentissage, tandis qu'un K trop grand peut réduire la performance du classificateur. Les expériences montrent que l'augmentation de K n'améliore pas toujours la précision [24].

1.3.2.2 Les Modèles d'apprentissage non supervisé

Les modèles d'apprentissage non supervisé comprennent diverses techniques conçues pour analyser des données non étiquetées et découvrir des motifs et des relations cachées. Parmi les modèles les plus importants, on trouve :

- **K-Means**

L'algorithme K-Means est l'une des techniques d'apprentissage non supervisé les plus connues. C'est un algorithme de regroupement qui organise des points de données similaires en clusters distincts. Les points de données appartenant au même cluster partagent des caractéristiques communes.

Comment fonctionne K-Means :

1. **Initialisation des centres des clusters (centroïdes) :** attribuer aléatoirement K centroïdes.
2. **Attribution des points de données :** chaque point est assigné au centroïde le plus proche en fonction de la distance (par exemple, la distance euclidienne).
3. **Recalcul des centroïdes :** calculer la position moyenne de tous les points dans chaque cluster et mettre à jour le centroïde en conséquence.
4. **Répéter jusqu'à convergence :** les étapes 2 et 3 sont répétées jusqu'à ce que les centroïdes ne changent plus de manière significative, ce qui signifie que les clusters sont stabilisés.

Ce processus garantit que des points de données similaires sont regroupés ensemble, améliorant ainsi la segmentation des données pour diverses applications telles que la segmentation des clients, la compression d'images et la détection des anomalies [25].

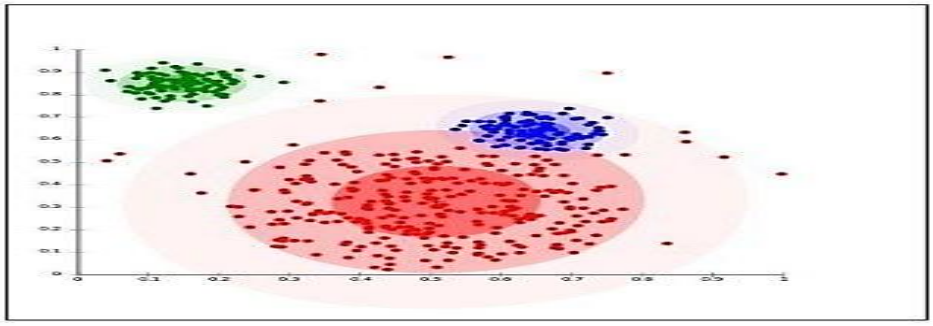


Figure.1.5. Exemple de regroupement K-Means [10]

Dans la figure 1.5, trois clusters peuvent être identifiés : un en rouge, un en bleu et un en vert [25].

- **Regroupement hiérarchique**

Le regroupement hiérarchique est une méthode qui construit une hiérarchie de clusters en fusionnant des clusters plus petits pour former des clusters plus grands (approche agglomérative) ou en divisant des clusters plus grands en plus petits (approche divisive), en fonction de leur similarité.

- **Règles d'association**

L'apprentissage des règles d'association est une technique d'apprentissage non supervisé utilisée pour identifier les relations entre les variables dans un jeu de données. Elle aide à découvrir des modèles cachés en déterminant la fréquence à laquelle certains éléments ou caractéristiques apparaissent ensemble. Cette méthode est largement appliquée dans divers domaines pour améliorer la prise de décision et l'efficacité opérationnelle [25].

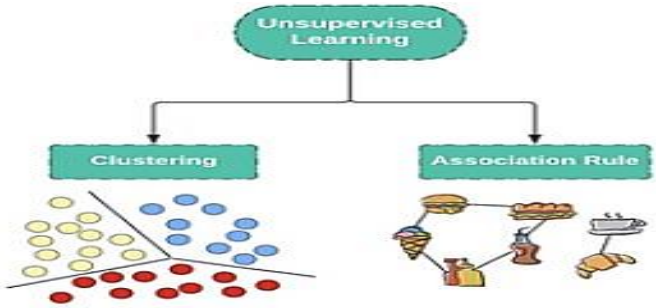


Figure 1.6 : Types d'apprentissage non supervisé – Regroupement et Règles d'association .

1.4 Apprentissage profond (Deep Learning)

L'apprentissage profond est un sous-ensemble de l'intelligence artificielle et de l'apprentissage automatique qui utilise des réseaux neuronaux comportant plusieurs couches pour apprendre automatiquement des modèles à partir des données. Il est largement utilisé dans la reconnaissance d'images, le traitement du langage naturel et l'analyse prédictive.

Différents types de réseaux neuronaux sont utilisés en apprentissage profond, chacun conçu pour des tâches spécifiques. Les sections suivantes couvriront les réseaux de neurones artificiels (ANNs) et leurs variantes, telles que les réseaux de rétropropagation (Back propagation Neural Networks), le perceptron monocouche (Single-layer Perceptron) et le perceptron multicouche (Multi-layer Perceptron). De plus, les réseaux de neurones convolutifs (CNNs), utilisés pour le traitement des images, et les réseaux de neurones récurrents (RNNs), spécialisés dans l'analyse des données séquentielles, seront également explorés.

1.4.1 Réseaux de neurones artificiels (ANNs)

Les réseaux de neurones artificiels (ANNs) sont des outils puissants en apprentissage automatique, largement utilisés pour des tâches telles que la reconnaissance d'images, le traitement du langage naturel et le jeu. Les ANNs apprennent à partir des données d'entraînement, ce qui les rend particulièrement efficaces pour les données non structurées où les relations entre les caractéristiques sont complexes. Cette section explore l'inspiration derrière les ANNs, leur fonctionnement et leur application pour résoudre divers problèmes.

Pour comprendre la place des ANNs dans le paysage plus large de l'apprentissage automatique, il est essentiel de revoir la structure et la catégorisation des algorithmes d'apprentissage automatique. L'apprentissage profond fait référence à un sous-ensemble de l'apprentissage automatique qui utilise des ANNs avec différentes architectures pour atteindre des objectifs spécifiques. L'apprentissage profond, y compris les ANNs, peut traiter des problèmes supervisés, non supervisés et par renforcement. La figure 1.7 illustre comment l'apprentissage profond est lié aux ANNs et à d'autres concepts d'apprentissage automatique.

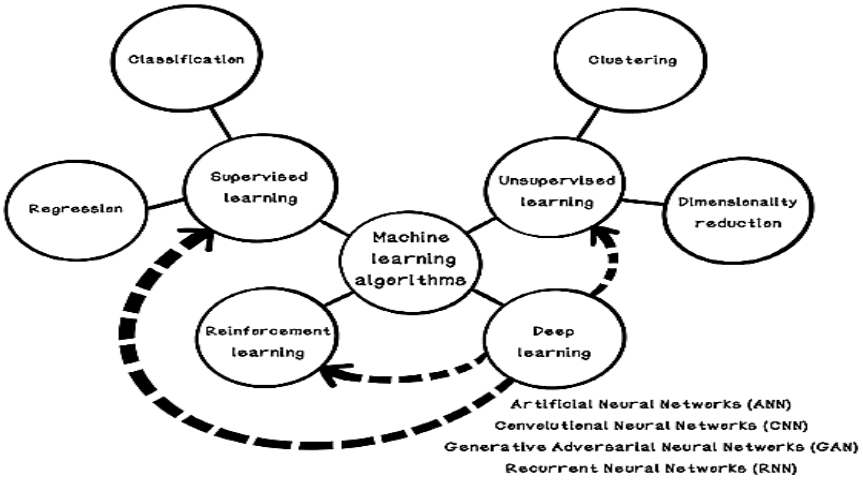


Figure 1.7 : Une carte décrivant la flexibilité de l'apprentissage profond et des réseaux de neurones artificiels (ANNs).

Les ANNs sont une composante essentielle du cycle de vie de l'apprentissage automatique, comme illustré dans la Figure 1.8. Ce cycle commence par l'identification d'un problème, la collecte et la compréhension des données pertinentes, la préparation du modèle, puis la phase de test et d'amélioration itérative pour optimiser les performances.

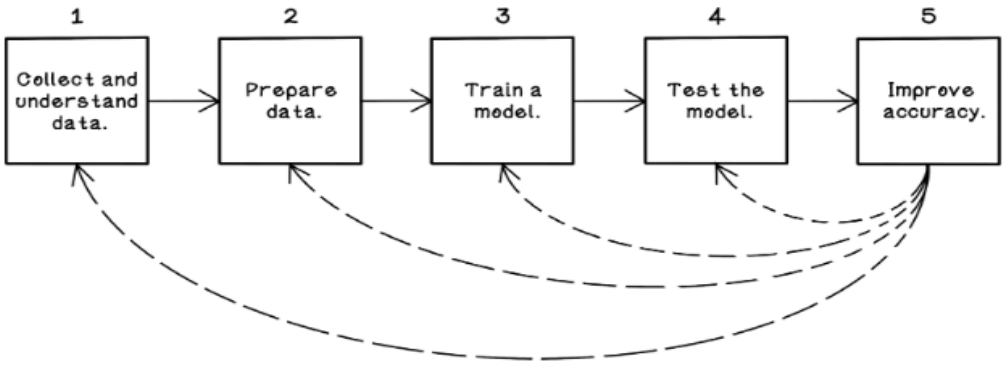


Figure 1.8 : Un flux de travail pour les expériences et projets d'apprentissage automatique.

Maintenant que nous comprenons comment les ANNs s'intègrent dans le cadre de l'apprentissage automatique, nous pouvons explorer leur intuition et leur fonctionnement.

Comme les algorithmes génétiques et les algorithmes d'intelligence collective, les ANNs sont inspirés par des phénomènes naturels dans ce cas, le cerveau humain et le système

nerveux. Le système nerveux permet les sensations et les fonctions cognitives, reposant sur des neurones qui communiquent par des signaux électriques et chimiques.

Les réseaux de neurones sont composés de neurones interconnectés qui transmettent et traitent des informations pour accomplir des tâches spécifiques. Prenons l'exemple d'un enfant apprenant à boire dans une tasse. Au début, il rencontre des difficultés et laisse souvent tomber la tasse.

Avec le temps, à force de répétitions, il apprend à tenir la tasse avec les deux mains, puis avec une seule, réussissant finalement à boire une gorgée. Ce processus d'apprentissage reflète l'entraînement des ANNs, où une exposition répétée et des ajustements améliorent les performances.

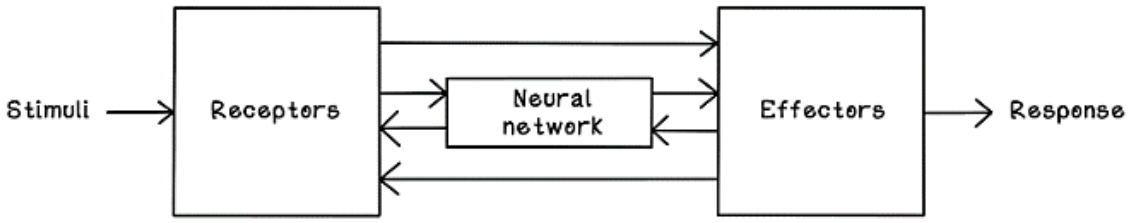


Figure 1.9 : Un modèle simplifié d'un système neuronal biologique.

Un neurone biologique, comme illustré dans la Figure 1.9, comprend plusieurs composants clés :

- Dendrites : Reçoivent les signaux des autres neurones.
- Corps cellulaire et noyau : Traitent et ajustent les signaux.
- Axone : Transmet les signaux à d'autres neurones.
- Synapses : Transfèrent et modifient les signaux avant de les transmettre aux dendrites du neurone suivant.

Avec environ 90 milliards de neurones travaillant ensemble, le cerveau humain fonctionne à un niveau d'intelligence extraordinaire.

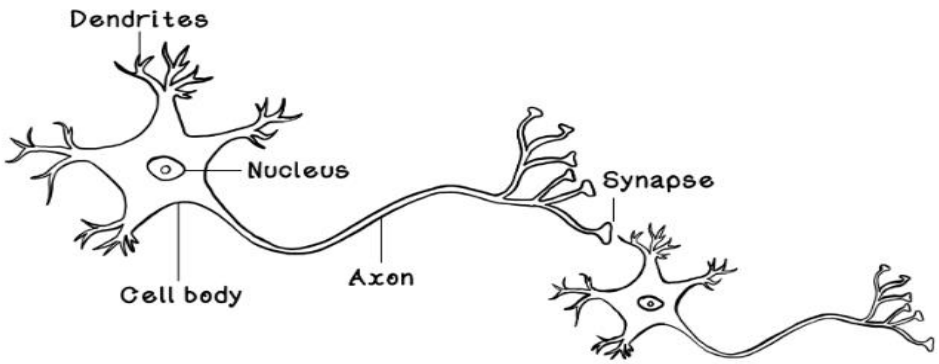


Figure 1.10 : La composition générale des neurones.

Bien que les réseaux de neurones artificiels (RNA) soient inspirés des réseaux neuronaux biologiques, ils ne sont pas des répliques directes du fonctionnement du cerveau humain. Alors que les RNA adoptent plusieurs principes observés dans les systèmes biologiques, les neurosciences continuent de découvrir les complexités du cerveau et du système nerveux [26].

1.4.2 Types de Réseaux de Neurones (RNNs)

Les réseaux de neurones existent sous différentes formes, chacun étant conçu pour traiter des tâches d'apprentissage spécifiques. Parmi eux, les réseaux de neurones récurrents (RNNs) sont spécialisés dans le traitement des données séquentielles. Différentes architectures ont été développées pour améliorer l'efficacité de l'apprentissage et résoudre divers défis. Les principaux types incluent :

1.4.2.1 Réseaux de Neurones à Rétropropagation

La rétropropagation est une méthode fondamentale pour l'entraînement des réseaux de neurones artificiels. Elle permet de minimiser la fonction de coût en ajustant les poids grâce à la descente de gradient. Cette technique calcule les dérivées partielles de la fonction de coût par rapport aux poids du réseau, permettant ainsi un apprentissage efficace.

Chaque couche du réseau se voit attribuer un terme d'erreur, calculé à partir de l'erreur de la couche suivante, introduisant ainsi le concept de « rétropropagation de l'erreur ». Dans la couche de sortie, l'erreur est directement dérivée de la différence entre la sortie prédite et les étiquettes réelles. Pour les couches cachées, l'erreur est obtenue en multipliant l'erreur de la couche suivante par la matrice de poids transposée et en appliquant la dérivée de la fonction d'activation.

Une fois les termes d'erreur déterminés, ils sont utilisés pour calculer les gradients des poids, essentiels pour la mise à jour des poids du réseau via la descente de gradient. Bien que la rétropropagation elle-même ne modifie pas les poids, elle fournit les informations nécessaires pour des ajustements efficaces des poids.

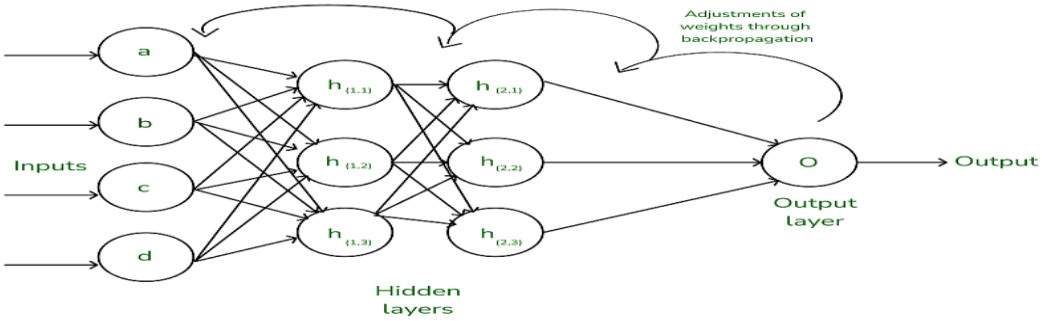


Figure 1.11 : Illustration simplifiée du fonctionnement de la rétropropagation

1.4.2.2 Modèle de Perceptron à Couche Unique

Le perceptron à couche unique, créé par Frank Rosenblatt, est la forme la plus simple d'un réseau de neurones. Il se compose d'un seul neurone qui traite les valeurs d'entrée, applique des poids, et passe le résultat à travers une fonction d'activation pour produire une sortie.

Une limitation majeure de ce modèle est qu'il ne peut résoudre que des problèmes linéairement séparables, ce qui signifie qu'il ne peut pas traiter des motifs complexes comme le problème XOR. Cette limitation a conduit au développement de réseaux de neurones plus avancés avec plusieurs couches.

L'image suivante illustre la structure d'un perceptron à couche unique :

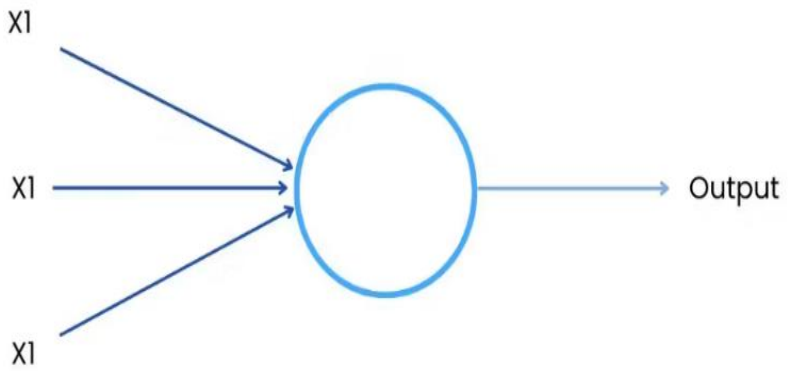


Figure 1.12 : Modèle de perceptron à couche unique

1.4.2.3 Modèle de Perceptron Multicouche

Pour surmonter les limitations des perceptrons à couche unique, les chercheurs ont développé des perceptrons multicouches (MLP). Ces réseaux sont composés de plusieurs couches :

- **Couche d'Entrée :** Accepte les données brutes (par exemple, des nombres, du texte, des images).
- **Couches Cachées :** Composées de plusieurs neurones qui appliquent des transformations à l'aide de fonctions d'activation non linéaires (telles que ReLU ou Sigmoid). Ces couches permettent au modèle d'apprendre des motifs complexes.
- **Couche de Sortie :** Produit la prédiction finale, que ce soit pour la classification (par exemple, identifier des emails spam) ou la régression (par exemple, prédire les prix des actions).

Les MLP utilisent des neurones sigmoid au lieu de perceptrons simples, leur permettant de traiter des relations non linéaires dans les données. Ils sont entraînés à l'aide d'une méthode appelée rétropropagation, qui ajuste les poids pour minimiser le serreur de prédiction [27].

L'image suivante présente une représentation visuelle d'un perceptron multicouche (MLP) :

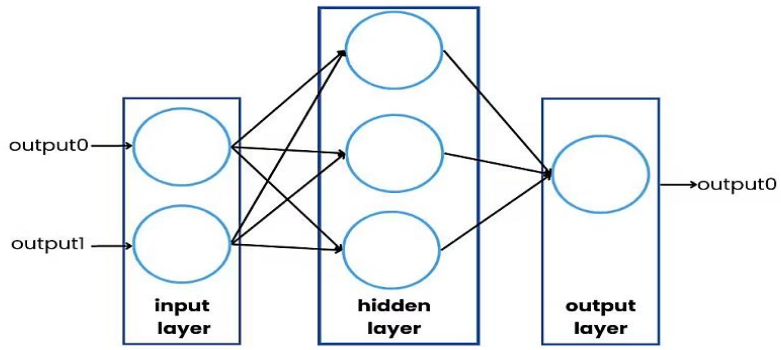


Figure 1.13 : Modèle de Perceptron Multicouche

1.4.3 Réseaux de Neurones Convolutionnels (CNN)

Les réseaux de neurones convolutionnels (CNN) sont un type spécial de réseaux de neurones à propagation avant. Ils sont similaires aux réseaux de neurones traditionnels, mais supposent que les entrées sont de type image, permettant ainsi d'encoder certaines propriétés

dans l'architecture. Les CNN utilisent des convolutions pour capturer l'invariance de translation, rendant la fonction de propagation avant plus efficace, réduisant le nombre de paramètres et simplifiant l'optimisation.

Les couches CNN sont structurées en dimensions telles que les canaux, la largeur, la hauteur et le nombre de filtres. Comme les perceptrons multicouches (MLP), les CNN se composent de couches séquentielles, y compris des couches convolutionnelles, des couches de pooling et des couches entièrement connectées.

La couche convolutionnelle applique un filtre (noyau) qui glisse sur l'entrée, calculant des cartes d'activation par des opérations de produit scalaire. Ce processus aide le réseau à apprendre à détecter des caractéristiques visuelles telles que les bords et les motifs à différents niveaux. Chaque couche contient plusieurs filtres, produisant des cartes d'activation séparées qui sont combinées pour former le volume d'activation final [28].

1.4.3.1 DenseNet-121

DenseNet-121 appartient aux Réseaux de Convolution Connectés de Manière Dense. Le concept le plus important de DenseNet est son bloc de connexion dense. Chaque couche reçoit des entrées supplémentaires de toutes les couches précédentes, puis transmet sa sortie sous forme de carte de caractéristiques aux couches suivantes. Il utilise la concaténation et chaque couche reçoit les connaissances collectives des couches précédentes.

Le concept est similaire aux ResNets, la seule différence étant que, plutôt que l'addition, cette couche effectue une concaténation des résultats des couches précédentes. DenseNet élimine de manière exceptionnelle le problème des gradients qui disparaissent, car il assure un flux maximal d'informations à travers le réseau [29].

DenseNet-121, avec 121 couches, est plus facile à entraîner et nécessite moins de mémoire. Les architectures CNN VGG-19, DenseNet-121, Inception v3, Inception-ResNet-v2 et Xception sont utilisées pour l'analyse de survie sur des données génomiques et histopathologiques.

1.4.3.2 U-Net

À un niveau élevé, U-Net se compose de deux parties principales un encodeur convolutionnel constitué de plusieurs couches convolutionnelles suivies de couches de pooling pour réduire la dimensionnalité ; et un décodeur symétrique constitué de couches de

transposition convolutionnelle (ou d'upsampling) permettant de reconstruire la carte de sortie avec la même résolution que l'entrée. L'architecture est résumée dans la Figure.1.14.

Les unités de base de chaque bloc convolutionnel sont une couche convolutionnelle, une fonction d'activation (généralement ReLU dans U-Net), et une opération de pooling (typiquement max-pooling). Il est à noter que contrairement à des architectures plus anciennes comme LeNet, U-Net utilise des techniques modernes telles que ReLU et le max-pooling pour de meilleures performances. Chaque couche convolutionnelle utilise typiquement un noyau de 3×3 et une fonction d'activation ReLU.

Figure. 1.14 Flux de données dans U-Net. L'entrée est une image (par exemple, de plante atteinte d'une maladie), la sortie est une carte de segmentation ou une probabilité par pixel appartenant à une classe cible.

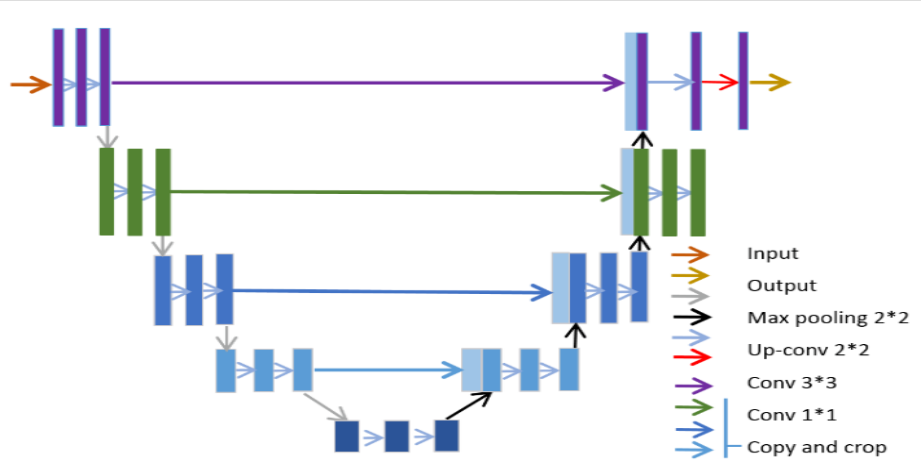


Figure 1.14 : Architecture de U-Net

Ces couches mappent les entrées disposées spatialement sur un certain nombre de cartes de caractéristiques bidimensionnelles, augmentant typiquement le nombre de canaux à mesure que l'on descend dans l'encodeur. L'encodeur contient généralement 4 à 5 niveaux, chaque niveau doublant le nombre de canaux.

Chaque opération de pooling 2×2 (stride 2) réduit la dimensionnalité par un facteur de 4 via un sous-échantillonnage spatial. À l'inverse, le décodeur effectue des opérations d'upsampling pour restaurer la taille initiale de l'image tout en fusionnant les caractéristiques issues de l'encodeur via des connexions de saut.

Avant la dernière couche de sortie, les cartes de caractéristiques du décodeur sont aplaties ou projetées sur une carte de sortie finale grâce à une couche convolutionnelle 1×1,

qui agit comme une couche de classification pixel-par-pixel. Cette couche finale possède autant de canaux de sortie que de classes de segmentation (ou catégories de maladies, par exemple), souvent suivie d'une fonction softmax pour interpréter les résultats comme des probabilités.

La sortie de U-Net est donc une carte de segmentation avec une dimension égale à celle de l'image d'entrée, où chaque pixel est associé à une probabilité d'appartenir à une classe. Comme nous effectuons une classification, la couche de sortie correspond au nombre de classes possibles [30].

1.4.4 Réseaux de Neurones Récurrents (RNNs)

Les réseaux de neurones récurrents (RNNs) diffèrent des réseaux de neurones artificiels (ANNs) en raison de leur architecture unique, qui inclut des boucles permettant à l'information de se propager à travers différents pas de temps. Chaque nœud est interconnecté non seulement au niveau du pas de temps actuel, mais aussi avec lui-même aux étapes de temps précédentes, formant une séquence. Cette structure temporelle permet aux RNNs de conserver des informations provenant des entrées précédentes, ce qui les rend adaptés au traitement des données séquentielles.

Dans les RNNs, chaque nœud possède une connexion de rétroaction, comme on le voit dans l'architecture d'Elman, où la sortie au temps t est influencée à la fois par l'entrée actuelle et la sortie précédente au temps t−1. Cette structure permet aux RNNs de capturer des dépendances au fil du temps, ce qui les rend efficaces pour des tâches impliquant des séries temporelles, la reconnaissance vocale et le traitement du langage naturel. Cependant, en raison de leur nature récurrente, les RNNs peuvent rencontrer des défis tels que les gradients qui disparaissent ou explosent, ce qui peut affecter l'apprentissage des dépendances à long terme [31].

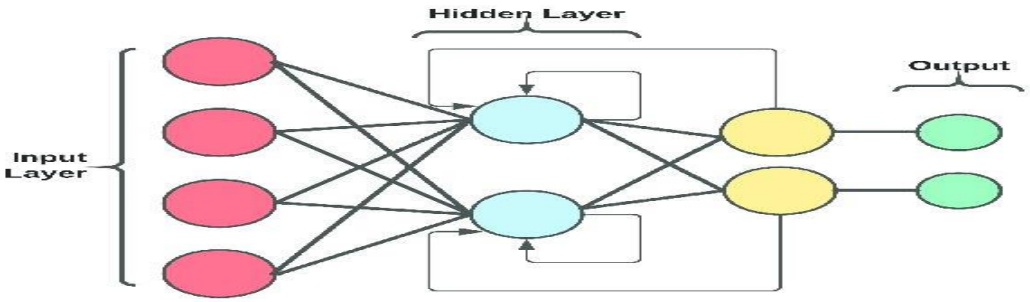


Figure 1.15 : Réseaux de neurones récurrents pour l'identification des maladies des plantes.

1.5 Apprentissage par Transfert

Lors de l'application de l'apprentissage automatique, on est limité par la quantité de données disponibles et la puissance de traitement. L'utilisation de l'apprentissage par transfert peut aider à atténuer ce problème.

L'apprentissage par transfert est une méthode utilisée pour améliorer la précision et réduire le temps d'entraînement global dans l'apprentissage automatique, en utilisant les connaissances acquises en résolvant une tâche pour résoudre une tâche similaire. L'impact de la performance de l'apprentissage par transfert dépend de la similarité entre les domaines source et cible.

Une grande similarité peut produire un gain de performance positif, tandis que le transfert de connaissances entre des domaines de faible similarité pourrait affecter négativement les performances d'un modèle. Intuitivement, extraire les paramètres appris d'un modèle et les implémenter dans un autre est une façon courante d'utiliser l'apprentissage par transfert.

Les paramètres appris sont générés en entraînant le modèle sur un ensemble de données différent de l'ensemble de données cible, un processus appelé préentraînement. Cela peut être couplé avec un ajustement ultérieur des paramètres pour affiner le modèle spécifiquement pour le domaine cible [32].

1.6 Conclusion

Ce chapitre a présenté une introduction à l'intelligence artificielle (IA) et à l'apprentissage automatique, en mettant en lumière les différentes approches permettant aux machines d'apprendre ou de fonctionner de manière intelligente. Il a abordé la distinction entre intelligence humaine et IA, le rôle central des algorithmes dans la prise de décision, ainsi que les principales techniques d'apprentissage supervisé, non supervisé et par renforcement. L'importance croissante des algorithmes dans l'amélioration des performances des systèmes d'IA a également été soulignée. Enfin, le chapitre a préparé le terrain pour l'étude des applications concrètes de l'IA, notamment dans la détection et la classification des maladies des plantes.

Chapitre 2 :

Détection et classification des maladies de plantes par apprentissage automatique

2.1 Introduction

Dans le contexte de l'agriculture moderne, la détection précoce des maladies des plantes devient cruciale afin que la production soit optimale et que les pertes économiques soient minimisées. Grâce à l'essor des techniques d'intelligence artificielle, systèmes d'apprentissage machine inclus, il est désormais possible de concevoir des systèmes automatisés capables de diagnostiquer très précisément et efficacement les maladies des plantes. Ce chapitre porte sur le renseignement et la classification sur le modèle de maladies des plantes selon l'apprentissage automatique. Nous examinerons les étapes majeures de ces systèmes, de l'acquisition d'images à la classification de maladies, à travers le prétraitement d'images et l'extraction de caractéristiques. Enfin, une revue des travaux antérieurs présentera les approches déjà existantes ainsi que leur contribution dans cette thématique.

2.2 Maladies des Plantes

Les maladies des plantes représentent un défi majeur dans le secteur agricole, affectant négativement la qualité et le rendement des cultures. Une maladie des plantes peut être définie comme toute condition nuisible qui affecte l'apparence ou le fonctionnement d'une plante. Les causes des maladies des plantes peuvent être classées en deux grandes catégories : les facteurs biotiques tels que les champignons, les bactéries, les virus et les nématodes, et les facteurs abiotiques tels que les extrêmes de température et les carences en nutriments.

Un concept fondamental en pathologie végétale est le Triangle des Maladies, qui stipule qu'une maladie ne peut se développer que lorsque trois éléments essentiels sont présents :

1. **Un hôte sensible :** Une plante vulnérable à l'infection.
2. **Un agent pathogène** : L'agent causal de la maladie.
3. **Un environnement favorable** : Des conditions environnementales qui favorisent le développement de la maladie.

Le processus d'infection commence lorsque le pathogène pénètre avec succès dans la plante et atteint des tissus spécifiques où l'infection peut se produire. Si les conditions environnementales sont favorables, le pathogène commence à se développer et à se propager, entraînant l'apparition de symptômes visibles de la maladie.

Les plantes réagissent aux maladies de trois manières principales :

1. **Surdéveloppement des tissus** : Formation de galles, gonflements ou enroulements des feuilles.
2. **Sous-développement des tissus** : Nanisme, chlorose (manque de chlorophylle) ou développement incomplet des organes.
3. **Mort des tissus** : Brûlure, taches foliaires, flétrissement et chancres [33].

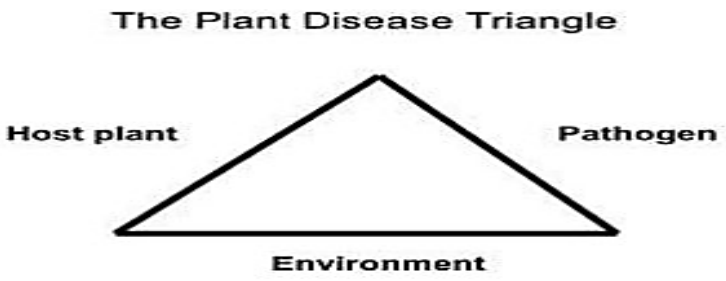


Figure 2.1 : le triangle des maladies des plantes

2.3 Systèmes de diagnostic des maladies agricoles

Parmi les définitions que nous avons trouvées dans la littérature, celle donnée par décrit succinctement ce qu'est un système de prédiction des maladies agricoles : " C'est un outil de gestion utilisé pour prédire l'apparition ou la détérioration des maladies des plantes cultivées ". Les producteurs utilisent ces systèmes pour prendre des décisions économiques sur les traitements de contrôle des maladies.

Les systèmes posent généralement aux producteurs une série de Questions sur la sensibilité de la culture hôte et font des re- commanditions en conjonction avec les conditions météorologiques actuelles et prévisibles. Généralement, les recommandations visent à déterminer le besoin pour le traitement de la maladie associée.

Selon cette définition, la tâche principale d'un système de diagnostic des maladies des plantes est de détecter de manière adéquate l'apparition de maladies à l'avance en passant par plusieurs étapes comme l'indique la figure 2.2, afin que les producteurs puissent prendre des décisions correctes sur l'application de produits phytosanitaires. Dans ce qui suit, nous verrons les bases, les conditions préalables et les processus des systèmes de diagnostic des maladies des plantes.

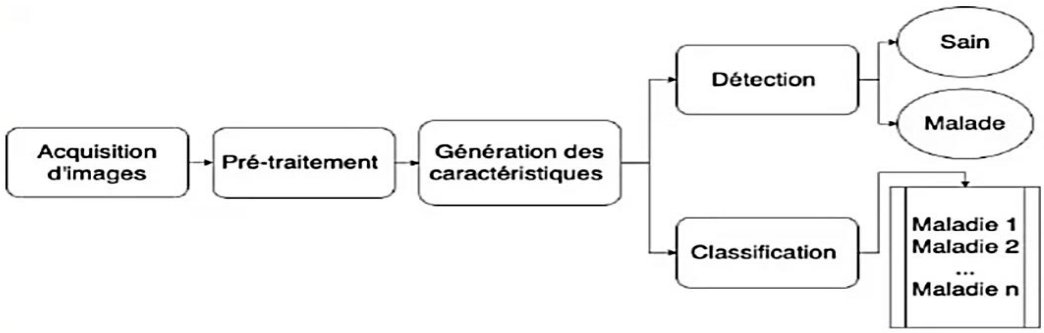


Figure 2.2 **:** Schéma synoptique d'un système de diagnostic des maladies des plantes

2.4 Étapes de détection des maladies agricoles

Tout système de vision par ordinateur passe par plusieurs étapes avant d'arriver à la solution souhaitée, comme l'indique le diagramme de la figure 2.3 :

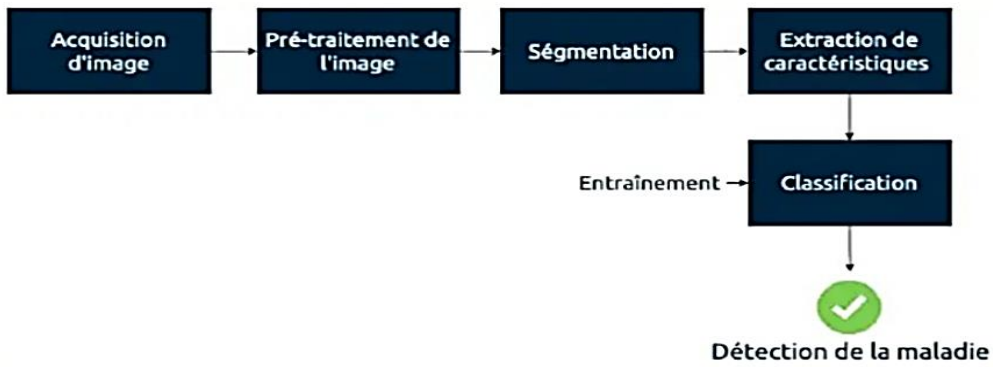


Figure 2.3: Schéma synoptique montrant les étapes de détection des maladies agricoles

2.4.1 Collecte de données(Acquisition des images)

Dans cette phase, des images pertinentes de l'objet sont capturées pour effectuer la classification à l'aide de méthodes automatisées. Les caméras numériques haute résolution et les smartphones sont utilisés pour enregistrer des échantillons d'images dans divers formats tels que jpg, png et tif [34]. Si les images collectées ne sont pas adéquates, des techniques d'amélioration sont appliquées [35].

L'acquisition d'images est cruciale pour une classification précise des maladies, car l'efficacité du modèle dépend de la qualité des données collectées [34].

2.4.2 Pré-traitement des images

Le prétraitement des images constitue une étape essentielle dans le domaine de la vision par ordinateur et du traitement des images. Cette phase initiale consiste à effectuer des opérations à un niveau d'abstraction minimal, où les données d'entrée et de sortie sont exclusivement des images [36]. En raison de la dégradation potentielle de la qualité des images causée par divers facteurs tels que les ombres, les distorsions, le bruit et les arrière-plans complexes, le prétraitement devient indispensable pour améliorer la qualité des images et les préparer pour les étapes de traitement ultérieures [37].

La majorité des ensembles de données utilisés sont souvent collectés dans des conditions réelles, ce qui peut introduire des informations indésirables. Par conséquent, avant l'extraction des caractéristiques, les images subissent un prétraitement visant à optimiser la précision des systèmes de détection des maladies des plantes. De plus, le recours à des opérations telles que le redimensionnement et le recadrage permet de réduire le temps de traitement global. Les techniques couramment utilisées dans le prétraitement peuvent être résumées comme suit :

a) Normalisation des images

Lorsque la méthode de caractérisation produit des descripteurs dépendant de la taille des images, il est nécessaire de procéder à une normalisation des tailles. Cette étape est cruciale pour garantir l'uniformité des données en vue de leur utilisation dans les méthodes de classification, qui requièrent des données de dimensions identiques. En outre, la normalisation peut également être employée pour réduire le volume des données, ce qui contribue à diminuer la complexité du traitement, comme illustré dans la figure 2.4.

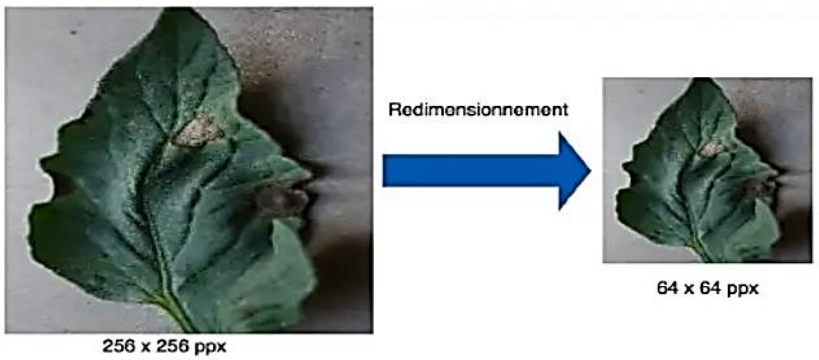


Figure 2.4**:** Redimensionnement d'une image

b) Suppression du bruit

Le prétraitement vise à éliminer le bruit généré par la scène, les conditions de prise de vue ou le capteur d'acquisition à travers l'application de filtres passe-bas. La figure 2.5 présente un exemple d'utilisation d'un filtre gaussien permettant de réduire le bruit additif, bien que cela puisse se faire au détriment des détails des contours.

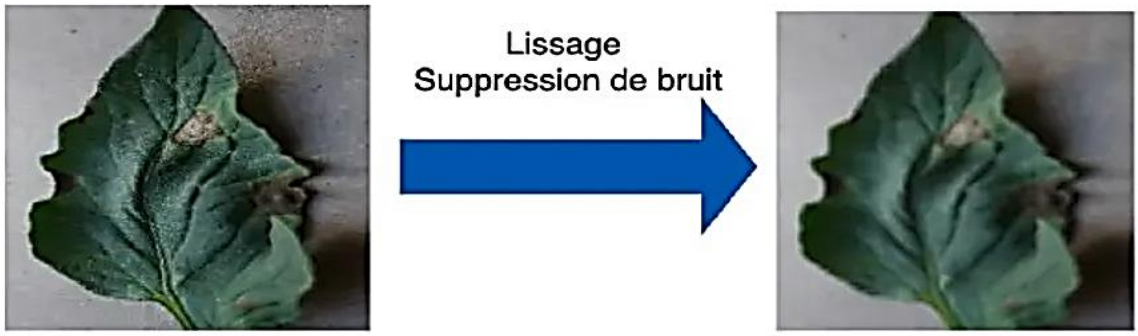


Figure 2.5: Lissage d'une image

c) Défection de contours

La détection des contours permet de mettre en évidence la forme d'une feuille, laquelle peut varier en fonction de l'état de santé des plantes. En vision par ordinateur, les détecteurs de contours reposent généralement sur le filtrage convolutif, qui met en exergue les zones présentant des variations marquées d'intensité. Parmi les techniques les plus utilisées, on retrouve les filtres de Sobel, Prewitt, La placien, Kirsch et Canny [38]. À titre d'exemple, la figure 2.6 illustre la détection des contours à l'aide des filtres de Sobel et du Laplacien à 4 connexions.

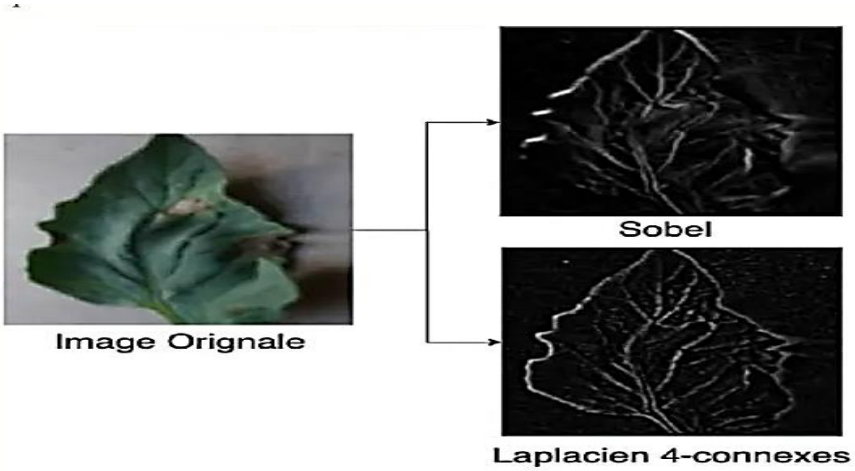


Figure 2.6**:** Détection des contours d'une image

d) Segmentation des images

La segmentation a pour objectif de diviser l'image afin d'identifier les régions d'intérêt, notamment celles présentant des anomalies. Elle permet d'isoler les zones affectées, offrant ainsi une représentation simplifiée et plus pertinente de l'image pour distinguer les régions saines des régions infectées. La figure 2.7 illustre deux exemples de segmentation appliquée à une plante infectée.

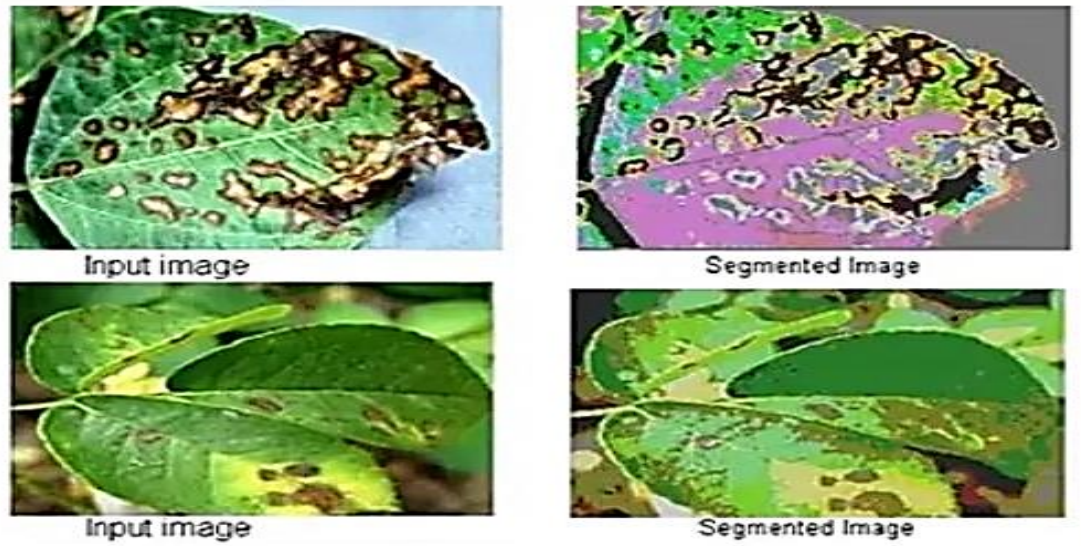


Figure 2.7: Exemples des images avant et après la segmentation [39]

2.4.3 Génération des caractéristiques

Dans le domaine de l'agriculture, le processus d'extraction de caractéristiques à partir de données brutes est appelé extraction de caractéristiques. Les descripteurs de caractéristiques des images d'entrée incluent les propriétés de forme, de couleur et de texture. Cela joue un rôle essentiel dans les tâches de classification. Dans le contexte de l'apprentissage automatique, l'ingénierie des caractéristiques est une technique fondamentale qui consiste à transformer des données brutes en un ensemble de caractéristiques significatives et pertinentes [40]. Le jeu de données est fourni comme entrée à cette étape pour déterminer si les plantes sont saines ou non.

Les caractéristiques de base dans une image incluent la couleur, la texture, la morphologie et d'autres propriétés connexes. Lorsqu'il s'agit d'identifier une tache sur une feuille endommagée, les traits morphologiques se révèlent plus efficaces que les autres [41]. Les caractéristiques de couleur telles que les moments de couleur et la texture de Gabor sont

fréquemment utilisées. Plusieurs méthodes permettent d'obtenir ces caractéristiques, comme l'histogramme de couleurs [42], le correlogramme de couleurs [43], le moment R de la couleur entre autres. Des éléments tels que le contraste, l'homogénéité, la variance et l'entropie peuvent également enrichir la texture.

Dans le contexte de l'identification des maladies des plantes, il a été constaté que l'utilisation des caractéristiques de texture produit des résultats plus favorables [44]. En utilisant la méthode de la matrice de cooccurrence de niveaux de gris (GLCM), on peut déterminer l'énergie, l'entropie, le contraste, l'homogénéité, le moment d'inertie et d'autres caractéristiques texturales [45-46]. Les caractéristiques texturales peuvent être séparées à l'aide de la transformation de Fourier (FT) et de la décomposition en paquets d'ondelettes [44].

D'autres caractéristiques telles que la fonction robuste d'accélération (SURF), l'histogramme des gradients orientés (HOG) et le Pyramid Histogram of Visual Words (PHOW) ont également démontré une plus grande efficacité.

2.4.4 Classification et détection de Maladie

L'identification et la classification des maladies des plantes utilisent la vision par ordinateur, un sous-domaine de l'Intelligence Artificielle (IA), qui permet aux machines d'imiter le système visuel humain. Cette technologie permet aux machines d'inspecter, de reconnaître et d'analyser des images du monde réel de manière similaire à la vision humaine. Dans le contexte de la détection des maladies des plantes, les techniques d'apprentissage automatique (ML) ont traditionnellement été utilisées pour la classification.

Cependant, avec les avancées de l'apprentissage profond (DL) un sous-ensemble du ML le potentiel d'amélioration de la précision est considérable. Plusieurs architectures DL développées, ainsi que diverses techniques de visualisation, ont été utilisées pour détecter et classifier les symptômes des maladies des plantes de manière plus efficace et précise [47].

Des secteurs tels que le diagnostic médical, l'espionnage, l'imagerie satellite et l'agro-industrie ont déjà démontré les avantages des technologies basées sur la vision par ordinateur. En agriculture, les systèmes équipés de vision par ordinateur peuvent être utilisés pour détecter et classifier les maladies des plantes en se basant sur des caractéristiques ou des symptômes spécifiques extraits des images des plantes.

Le processus de détection et de classification implique généralement une série d'étapes bien définies, débutant par l'acquisition d'images, suivie des tâches de traitement d'images telles que le redimensionnement, le filtrage, la segmentation, l'extraction et la sélection des caractéristiques. Enfin, la classification et la détection des maladies sont réalisées à l'aide de techniques d'apprentissage automatique ou d'apprentissage profond [48], [49].

2.5 Travaux connexes

Cette section présente les dernières recherches sur l'application de l'apprentissage profond à la détection des maladies des plantes, en particulier en utilisant l'ensemble de données Plant Village.

Les auteurs de [50] ont développé un système de détection des maladies virales des plantes, utilisant un ensemble de 800 images de feuilles de concombre. Leur modèle CNN a atteint une précision de 94,9% avec une validation croisée à 4 volets. Dans une autre étude [51], ils ont détecté sept types de maladies virales du concombre avec un ensemble de 7250 images, obtenant une précision de 82,3%.

Les auteurs de [52] ont utilisé un CNN pour reconnaître 13 types de maladies végétales à partir de 3000 images de feuilles. Leur modèle pré-entraîné CaffeNet a atteint une précision globale de 96,3%.

Une étude de [53], utilisant l'ensemble de données Plant Village (54 300 images), a permis de détecter 14 espèces de cultures et 26 maladies. Leur modèle basé sur Google Net a atteint une précision de 99,3%, mais a chuté à 31,4% lors des tests sur des images téléchargées en ligne.

Les auteurs de [54] ont utilisé l'architecture LeNet pour détecter des maladies sur des feuilles de bananiers, atteignant une précision de 99,72% avec un ensemble de 3700 images.

Enfin, une autre étude [29] a porté sur la détection de la gravité de la maladie du black rot des pommes. Avec un ensemble de plus de 150 images, leur modèle CNN (VGG16, VGG19, Inception-V3, ResNet50) a atteint une précision maximale de 90,4% avec VGG16.

Les articles présentés sont classés en fonction du type d'aliment étudié ou de l'ensemble de données utilisé. Les principaux aliments abordés vont de la tomate au riz, incluant des aliments régionaux comme le manioc. La tomate est la plus étudiée en raison de

son importance et de la variété de maladies qu'elle subit. Certains articles ont tenté de classer jusqu'à 38 maladies et espèces différentes [56] [57] [58].

L'ensemble de données le plus utilisé est Plant Village, disponible publiquement, tandis que d'autres ensembles privés ont été créés sans être rendus accessibles à la communauté de recherche. La majorité des recherches ont utilisé des modèles d'apprentissage profond pour classer les maladies à partir d'images, avec quelques rares études se concentrant sur la segmentation des zones malades ou l'estimation de la gravité de la maladie.

Certains articles ont exploré l'utilisation de modèles sur des appareils mobiles, ce qui est crucial pour une application en temps réel, surtout dans les pays en développement où l'accès à l'équipement et à internet est limité. Un modèle léger, efficace et entièrement déployé sur des dispositifs mobiles est donc une solution optimale.

2.6 Conclusion

Dans ce chapitre, l'utilisation des techniques d'apprentissage automatique pour la détection et la classification des maladies des plantes a été abordée en détail. Les différentes étapes, allant de l'acquisition des images à la classification des maladies, montrent l'importance de chaque phase pour garantir l'efficacité des systèmes de diagnostic. Bien que des progrès notables aient été réalisés, notamment en ce qui concerne la précision des modèles, des défis demeurent concernant l'amélioration de leur robustesse et la gestion de la variabilité des environnements agricoles. L'intégration de nouvelles sources de données et le développement de modèles légers, déployables sur des appareils mobiles, constituent des pistes de recherche prometteuses pour rendre ces technologies plus accessibles et efficaces, en particulier dans les pays en développement où l'accès à l'équipement et à internet peut être limité. Les travaux futurs devront donc se concentrer sur l'optimisation des modèles afin de les rendre plus généralistes et adaptables à divers contextes agricoles.

Chapitre 3 :

Un Système Basée CNN Pour la Reconnaissance des Maladies de la Pomme de Terre

3.1 Introduction

Dans ce dernier chapitre, nous présentons la mise en œuvre d'une application web interactive permettant de détecter et de classer automatiquement les maladies affectant les feuilles de pomme de terre, à l'aide de modèles d'apprentissage profond. L'objectif principal est de fournir une solution intelligente, accessible et rapide qui puisse être utilisée par les agriculteurs ou les spécialistes pour faciliter le diagnostic phytosanitaire. Le système repose sur une architecture moderne : le backend est conçu avec FastAPI, intégrant deux modèles performants, DenseNet et U-Net, préalablement entraînés sur des images de feuilles de pomme de terre atteintes de diverses maladies. L'interface utilisateur, réalisée en ReactJS, permet à l'utilisateur de téléverser une image, d'obtenir une prédiction immédiate et de visualiser le résultat de manière claire et intuitive.

3.2. Contexte de l'étude

La détection précoce des maladies de la pomme de terre est cruciale pour assurer la sécurité alimentaire et réduire les pertes agricoles. Dans ce contexte, les approches basées sur le Deep Learning offrent des solutions prometteuses, mais le choix de l'architecture optimale reste un défi. Cette étude vise à comparer systématiquement deux modèles profonds - U-Net et DenseNet121 - afin d'évaluer leurs performances respectives pour la segmentation et la classification des maladies, tout en identifiant leurs avantages complémentaires pour des applications agricoles pratiques.

Plusieurs travaux récents ont exploré l'utilisation des réseaux de neurones profonds pour la détection des maladies des plantes, avec des approches variées selon la nature du problème. Pour les tâches de segmentation sémantique des lésions foliaires, U-Net s'est imposé comme une référence, notamment grâce à sa capacité à capturer des détails fins grâce à ses connexions résiduelles entre l'encodeur et le décodeur [59]. Des études comme celle de [60] sur PlantVillage ont montré que les architectures de type CNN telles que DenseNet121 excellent dans la classification d'images de plantes malades, avec des taux de précision dépassant 99% dans des conditions contrôlées. Cependant, U-Net offre une granularité supérieure pour quantifier l'étendue des dommages, comme l'ont démontré [61] dans leur travail sur la segmentation des maladies de la tombe et de la pomme de terre. À l'inverse, DenseNet121, avec ses connexions denses permettant une réutilisation optimale des caractéristiques [62], s'avère plus efficace pour discriminer des pathologies visuellement similaires (mildiou vs alternariose). Une étude comparative récente de [63] a révélé que si

DenseNet121 atteint 98,7% de précision en classification, U-Net fournit des cartes de segmentation avec un IoU de 0,89, soulignant leur complémentarité. Ces résultats suggèrent que le choix entre les deux modèles doit s'appuyer sur l'objectif précis : diagnostic rapide (DenseNet121) ou analyse quantitative des lésions (U-Net), avec des possibilités d'intégration dans des pipelines hybrides pour des systèmes de monitoring agricole complets.

Partie I : Conception et Développement du Système

Cette partie présente la méthodologie suivie pour concevoir et développer un système de détection des maladies des plantes à l'aide de techniques d'apprentissage profond. Elle décrit dans un premier temps la base de données utilisée ainsi que le processus de préparation des données. Ensuite, elle expose les étapes de développement du système, tant du côté serveur (backend) que du côté utilisateur (frontend). Enfin, une attention particulière est accordée à l'évaluation du modèle de classification, à travers plusieurs métriques standards permettant de mesurer sa performance.

3.3 Source de données : Plant Village

Dans le cadre de ce projet, nous avons utilisé la base de données Plant Village, largement reconnue dans le domaine de la vision par ordinateur appliquée à la détection des maladies des plantes. Cette base propose un ensemble d'images annotées avec précision, collectées dans des conditions contrôlées pour entraîner des modèles d'intelligence artificielle.

L'étude s'est focalisée exclusivement sur des images de feuilles de pomme de terre (potato), totalisant 890 images réparties en trois classes distinctes :

- **Potato_Early_blight :** feuilles atteintes de la brûlure précoce.
- **Potato_Late_blight :** feuilles atteintes de la brûlure tardive.
- **Potato_healthy :** feuilles saines, sans aucune maladie apparente.

Les données ont été stockées dans un répertoire sur Google Drive, puis montées dans l'environnement de travail Google Colab à l'aide de la commande suivante :

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Le chargement des images s'est effectué via la fonction image_dataset_from_directory de la bibliothèque TensorFlow, avec des paramètres de taille fixés à 256x256 pixels et un batch size de 32, tout en activant le mélange aléatoire pour assurer une bonne généralisation du modèle :

```
[ ]  dataset = tf.keras.preprocessing.image_dataset_from_directory(
         "/content/drive/MyDrive/Colab Notebooks/potato-disease/PlantVillage",
         seed=123,
         shuffle=True,
         image_size=(IMAGE_SIZE,IMAGE_SIZE),
         batch_size=BATCH_SIZE
     )
```

```
Found 890 files belonging to 3 classes.
```

Les classes ont été automatiquement détectées à partir de la structure des sous-dossiers du répertoire, comme le montre la variable class_names :

```
[ ]  class_names = dataset.class_names
     class_names
```

```
['Potato___Early_blight', 'Potato___Late_blight', 'Potato___healthy']
```

La qualité et la structuration rigoureuse de cette base de données ont permis d'entraîner le modèle de manière efficace, en facilitant l'apprentissage supervisé des différentes maladies ciblées.

3.4. Architecture des modèles DenseNet121 et U-Net

Les modèles ResNet121 et U-Net représentent deux architectures profondes distinctes mais complémentaires en vision par ordinateur. **ResNet121**, basé sur les *residual networks* (réseaux résiduels), se caractérise par ses connexions skip qui contournent des couches pour éviter la disparition du gradient, ce qui en fait un choix robuste pour la classification d'images. À l'inverse, **U-Net**, avec sa structure en forme de U et ses connexions de skip entre l'encodeur et le décodeur, est spécialement conçu pour la segmentation sémantique, permettant une localisation précise des objets en combinant informations contextuelles et détails spatiaux. Les figures (figure 3.1 et figure 3.2) ilustreent les architectures consécutives des modeles modèle ResNet121et U-Net qu'on a adopté dans notre étude.

Figure 3.1 : Architecture du modèle ResNet121

3.4.1. Explications de l'architecture DenseNet121

- Resizing & Rescaling (Preprocessing) Cette couche prépare l'image en modifiant sa taille et en normalisant les valeurs des pixels. Cette étape est essentielle pour garantir que les données entrent avec la forme et la taille appropriées.
- Input Image (3×256×266) Image d'entrée de base avec 3 canaux de couleur (RGB) et une taille de 256×266 pixels. Elle représente les données brutes qui seront traitées à travers les couches du réseau neuronal.

- Dense Block 1 (6 layers) Bloc dense contenant 6 couches densément connectées où chaque couche reçoit les sorties de toutes les couches précédentes. Il aide à extraire les caractéristiques de base de l'image et améliore le flux d'informations.
- Transition Layer 1 (Conv + Pool) Couche de transition composée d'une convolution suivie d'un pooling (regroupement). Elle réduit les dimensions des données tout en préservant les caractéristiques importantes pour l'efficacité computationnelle.
- Dense Block 2 (12 layers) Bloc dense plus large avec 12 couches pour extraire des caractéristiques plus complexes des données traitées. Il augmente la profondeur du réseau et sa capacité à comprendre les détails complexes de l'image.
- Transition Layer 2 (Conv + Pool) Deuxième couche de transition appliquant les mêmes opérations (Convolution + Pooling) pour réduire les dimensions. Elle prépare les données pour l'étape suivante tout en préservant les informations essentielles.
- Dense Block 3 (24 layers) Le plus grand bloc dense du réseau avec 24 couches pour extraire les détails les plus fins et les caractéristiques complexes. Il représente la partie la plus profonde du réseau pour l'analyse détaillée des motifs.
- Transition Layer 3 (Conv + Pool) Troisième couche de transition qui continue la réduction dimensionnelle et la préparation des données. Elle maintient l'équilibre entre la complexité des caractéristiques et l'efficacité computationnelle.
- Dense Block 4 (16 layers) Dernier bloc dense avec 16 couches pour finaliser l'extraction des caractéristiques hauts niveaux. Il consolide toutes les informations apprises pour préparer la classification finale.
- Global Average Pooling Technique de regroupement qui calcule la moyenne de chaque carte de caractéristiques. Elle réduit drastiquement le nombre de paramètres et évite le surapprentissage.
- Dense Layer (64 neurons, ReLU) Couche entièrement connectée avec 64 neurones utilisant la fonction d'activation ReLU. Elle traite les caractéristiques extraites pour préparer la décision de classification.
- Output Layer (3 classes, Softmax) Couche de sortie finale avec 3 neurones utilisant la fonction Softmax pour la classification. Elle produit les probabilités pour chacune des 3 classes possibles, la somme étant égale à 1.

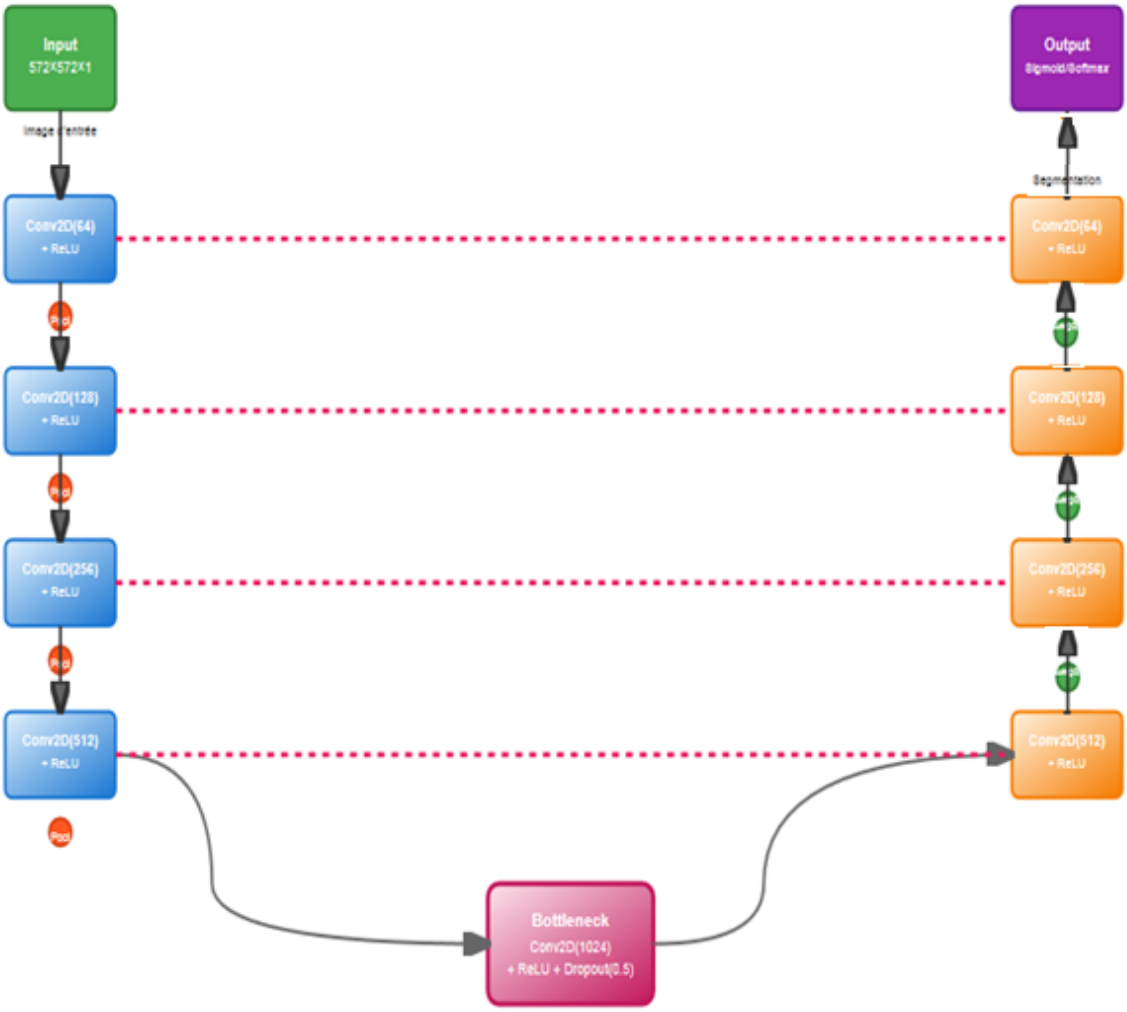


Figure 3.2 : Architecture du modèle U-net

3.4.2. Explications de l'architecture U-net

- Input Image (572×572×1) Image d'entrée en niveaux de gris avec une résolution de 572×572 pixels et un seul canal. Cette architecture est particulièrement adaptée aux images médicales et à la segmentation biomédicale
- Conv2D (64) + ReLU Activation Première couche de convolution avec 64 filtres suivie d'une activation ReLU pour extraire les caractéristiques de base. Cette couche détecte les contours simples et les textures élémentaires dans l'image d'entrée
- MaxPooling2D + Conv2D (128) + ReLU Réduction de la résolution spatiale par pooling max et augmentation du nombre de canaux à 128. Cette étape capture des caractéristiques plus abstraites tout en réduisant la dimension spatiale

- MaxPooling2D + Conv2D (256) + ReLU Continuation de l'encodage avec 256 filtres pour capturer des motifs plus complexes et contextuels. La résolution continue de diminuer tandis que la profondeur sémantique augmente
- MaxPooling2D + Conv2D (512) + ReLU Niveau d'encodage plus profond avec 512 filtres pour une représentation très abstraite des caractéristiques. Cette couche capture des contextes globaux et des relations spatiales à long terme
- Bottleneck: MaxPooling2D + Conv2D (1024) + ReLU + Dropout (0.5) Couche la plus profonde avec 1024 filtres et dropout pour éviter le surapprentissage. Elle représente la compréhension la plus abstraite de l'image avec régularisation par dropout
- UpSampling2D + Conv2D (512) + ReLU Première étape de décodage qui augmente la résolution spatiale tout en réduisant les canaux à 512. Elle commence à reconstruire les détails spatiaux perdus lors de l'encodage
- UpSampling2D + Conv2D (256) + ReLU Continuation du décodage avec fusion des informations des skip connections pour restaurer les détails. Cette étape combine les caractéristiques de haut niveau avec les détails de bas niveau
- UpSampling2D + Conv2D (128) + ReLU Poursuite de la reconstruction spatiale avec 128 filtres et intégration des skip connections. Elle affine progressivement la segmentation en utilisant les informations multi-échelles
- UpSampling2D + Conv2D (64) + ReLU Dernière étape de décodage avant la sortie, ramenant le nombre de canaux à 64. Cette couche finalise les détails fins de la segmentation avec une résolution proche de l'entrée
- Skip Connections (Copy & Crop) Connexions directes entre les couches de l'encodeur et du décodeur au même niveau de résolution. Elles permettent de préserver les détails fins perdus lors du downsampling et améliorent la précision de segmentation
- Output Layer (Conv2D + Sigmoid/Softmax) Couche de sortie finale qui produit la carte de segmentation avec le nombre de classes désiré. Utilise Sigmoid pour la segmentation binaire ou Softmax pour la segmentation multi-classes.

3.5 Évaluation du modèle de classification

C'est une étape cruciale pour évaluer la performance et la fiabilité du modèle. Voici quelques métriques couramment utilisées pour évaluer un modèle de classification.

3.5.1 La matrice de confusion

Considérons un classifieur binaire qui prédit deux classes (classe 0 et classe 1). On distingue alors quatre cas :

- **Vrai positif (VP) :** Élément de la classe 1 correctement prédit.

- **Vrai négatif (VN) :** Élément de la classe 0 correctement prédit.

- **Faux positif (FP) :** Élément de la classe 0 incorrectement prédit comme classe 1.

- **Faux négatif (FN) :** Élément de la classe 1 incorrectement prédit comme classe 0.

Ces informations sont présentées dans une matrice de confusion.

		Classe prédite	
		Classe0	*Classe1*
Classe réele	*Classe0*	*VN*	*FN*
	Classe1	*FP*	*VP*

Tableau 3.1 : Les informations des classes.

Si cette matrice est diagonale, cela signifie que le classifieur est parfait. La matrice de confusion peut aussi être généralisée à k classes (k > 2). Plusieurs indicateurs peuvent ensuite en être dérivés.

3.5.2 Métriques d'évaluation du modèle

3.5.2.1. L'exactitude (Accuracy)

Elle mesure la proportion de bonnes prédictions sur l'ensemble des prédictions.

$$\textbf{Exactitude} = \frac{TP+TN}{TP+TN+FP+FN} \quad \ldots\ldots\ldots\ldots (3.1)$$

3.5.2.2 La précision (Precision)

Elle mesure la proportion de vrais positifs parmi tous les éléments prédits comme positifs.

$$\textbf{Précision} = \frac{TP}{TP+FP} \quad \ldots\ldots\ldots\ldots (3.2)$$

3.5.2.3 Le rappel (Recall)

Le rappel correspond à la proportion de vrais positifs parmi tous les éléments réellement positifs.

$$\textbf{Rappel} = \frac{TP}{TP+FN} \quad \ldots\ldots\ldots\ldots (3.3)$$

3.52.4 Le F1-score

Le F1-score est la moyenne harmonique entre la précision et le rappel. Il est particulièrement utile lorsque le jeu de données est déséquilibré.

$$\textbf{F1-score} = \frac{2\times TP}{2\times TP+FP+FN} \quad \ldots\ldots\ldots\ldots (3.4)$$

3.5.3 Tests

Maintenant que notre application de détection des maladies des plantes est opérationnelle, nous passons à la phase de tests afin de vérifier son bon fonctionnement. Pour cela, il est nécessaire d'exécuter simultanément le fichier main.py de notre backend, ainsi que notre interface utilisateur (frontend) à l'aide de la commande suivante :

```
npm run start
```

Après l'exécution de cette commande, la fenêtre de l'interface utilisateur s'ouvre automatiquement dans le navigateur, comme illustré dans la figure 3.4 :

Figure 3.3: Page principale de notre interface web

Après avoir chargé une image d'une feuille de plante dans l'application on aura notre diagnostic avec un pourcentage de confiance, selon l'état de la feuille comme indique la figure 3.5:

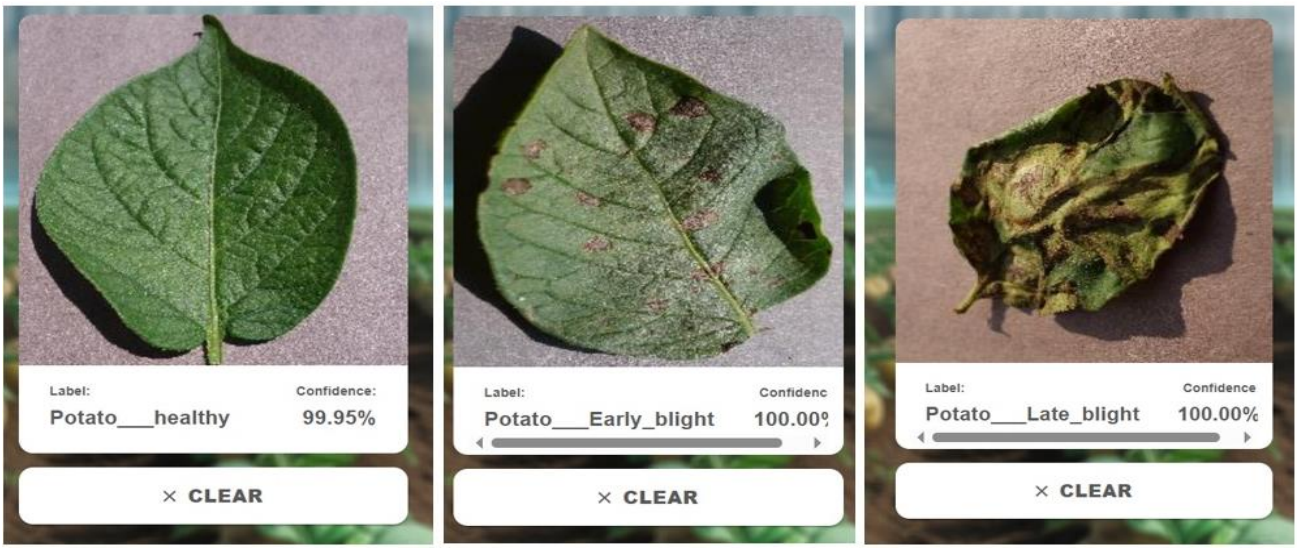


Figure 3.4: Réponse de L'application après L'exécution

Partie II : Résultats et Discussion

Cette partie présente l'analyse des résultats expérimentaux obtenus à partir des modèles d'apprentissage profond DenseNet121 et U-Net, appliqués à la classification des maladies de la pomme de terre à partir d'images de feuilles. Elle vise à évaluer la performance des modèles et la précision de la classification en s'appuyant sur plusieurs métriques standards.

3.6 Résultats et Analyse

3.6.1 Analyse des résultats de l'entraînement avec DenseNet121

Le modèle DenseNet121, basé sur une architecture profonde et exploitant l'apprentissage par transfert (Transfer Learning), a été entraîné sur une base de données contenant trois classes : Early Blight, Late Blight et feuille saine. L'entraînement s'est déroulé sur 50 époques avec une taille de lot fixe et un prétraitement standardisé des images.

L'analyse des indicateurs montre ce qui suit :

- **Fonction de perte (loss)** : elle a diminué progressivement de 0.1445 à 0.0194, indiquant une bonne convergence du modèle.
- **Précision d'entraînement (accuracy)** : elle a atteint 99,48 %.
- **Précision de validation (val_accuracy)** : elle a atteint 100 %, ce qui indique une excellente capacité de généralisation.

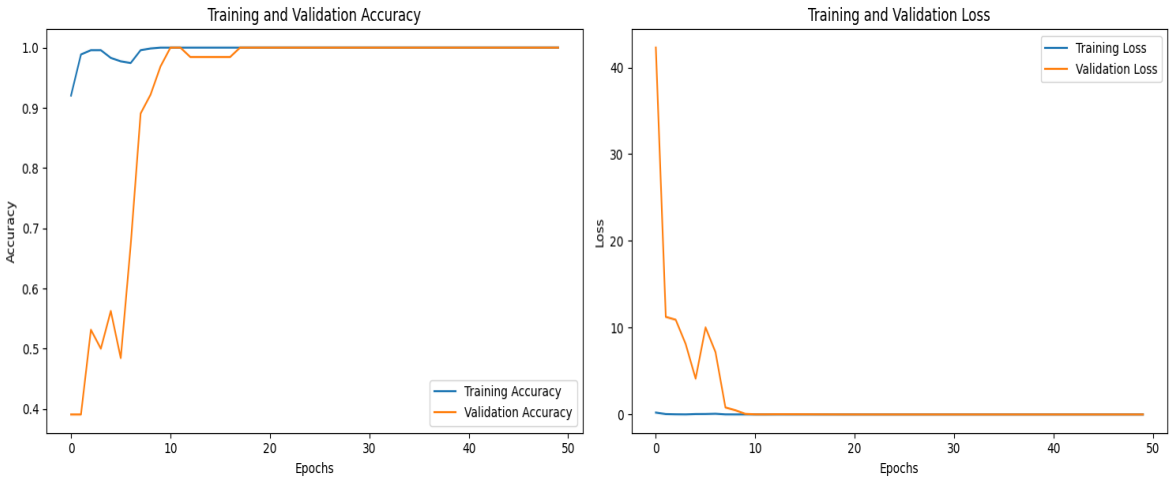


Figure 3.5 : Évolution de la précision et de la perte pour DenseNet121

3.6.2 Résultats de l'entraînement avec le modèle U-Net

Le modèle U-Net, reconnu pour son efficacité dans le traitement des images médicales et agricoles, possède une architecture qui permet de préserver les détails fins durant l'apprentissage. Il a été adapté à la classification des maladies de la pomme de terre dans les mêmes conditions d'entraînement.

- Loss : a diminué jusqu'à 0.0228, indiquant une bonne convergence.
- Accuracy d'entraînement : 98,7 %, et val_accuracy : 97,4 %.

Les courbes d'apprentissage ont montré une stabilité satisfaisante malgré quelques fluctuations modérées au milieu de l'entraînement.

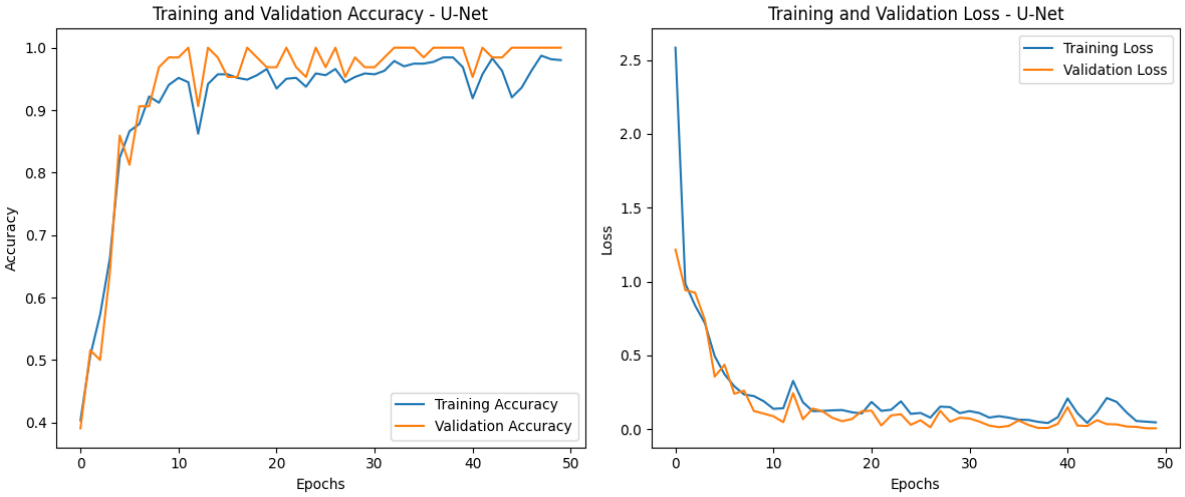


Figure 3.6 : Évolution de la précision et de la perte pour U-Net

3.6.3 Évaluation sur le jeu de données test

Les résultats expérimentaux obtenus sur le jeu de test permettent d'évaluer la performance finale des modèles DenseNet121 et U-Net dans un contexte réel de classification des maladies de la pomme de terre. Le tableau suivant résume la précision et la perte (loss) pour chaque modèle :

Modèle	Précision test	Perte test
DenseNet121	100 %	0.0007
U-Net	96,52 %	0.0371

Tableau 3.2 : Résultats de précision et de perte sur le jeu de test pour les modèles DenseNet121 et U-Net

On observe que le modèle DenseNet121 surpasse le modèle U-Net, avec une précision parfaite de 100 % et une perte extrêmement faible. Cela reflète une excellente généralisation du modèle sur les données de test, sans surapprentissage apparent. En revanche, U-Net affiche une performance légèrement inférieure, avec une précision de 96,52 %, ce qui reste néanmoins très satisfaisant.

3.6.3.1 Résultats du modèle DenseNet121

Classe	Précision	Rappel	F1-score	Support
Potato___Early_blight	1.00	0.98	0.99	43
Potato___Late_blight	0.98	1.00	0.99	61
Potato___healthy	1.00	1.00	1.00	18
Moyenne (macro)	0.99	0.99	0.99	122
Moyenne pondérée	0.99	0.99	0.99	122
Exactitude	/	/	0.99	122

Tableau 3.3 : Métriques de performance (précision, rappel, F1-score) du modèle DenseNet121 sur le jeu de test

Le tableau des résultats de DenseNet121 montre des performances quasi parfaites dans la classification des images des feuilles de pomme de terre :

- Pour la classe Potato___Early_blight, la précision est de 1.00, le rappel de 0.98, et le f1-score de 0.99, ce qui signifie que presque tous les échantillons atteints par le mildiou précoce ont été correctement détectés, avec très peu de faux positifs.
- Pour la classe Potato___Late_blight, le modèle atteint une précision de 0.98 et un rappel parfait de 1.00, ce qui signifie qu'il a identifié toutes les vraies occurrences de la maladie, avec une excellente fiabilité.
- La classe Potato___healthy obtient des scores parfaits (1.00) pour tous les indicateurs, ce qui reflète une capacité exceptionnelle à reconnaître les feuilles saines.
- Les moyennes macro et pondérée des scores (précision, rappel, f1-score) atteignent 0.99, ce qui confirme une homogénéité dans les performances sur toutes les classes.
- Enfin, l'exactitude globale (accuracy) est de 0.99, ce qui indique une excellente performance sur l'ensemble du jeu de test.

3.6.3.2 Résultats du modèle UNet

Classe	Précision	Rappel	F1-score	Support
Potato___Early_blight	0.96	1.00	0.98	43
Potato___Late_blight	1.00	0.89	0.94	61
Potato___healthy	0.78	1.00	0.88	18
Moyenne (macro)	0.91	0.96	0.93	122
Moyenne pondérée	0.95	0.94	0.94	122
Exactitude	/	/	0.94	122

Tableau 3.4 : Métriques de performance (précision, rappel, F1-score) du modèle U-Net sur le jeu de test

Le tableau correspondant à UNet révèle des résultats bons mais moins solides que ceux du modèle précédent :

- La classe Potato___Early_blight est bien reconnue avec une précision de 0.96 et un rappel de 1.00, ce qui signifie que toutes les vraies instances ont été détectées, bien que quelques erreurs de classification aient été commises (faux positifs).
- Pour Potato___Late_blight, bien que la précision soit parfaite (1.00), le rappel tombe à 0.89, ce qui indique que le modèle a manqué certaines images atteintes par cette maladie (faux négatifs).
- La classe Potato___healthy présente une faiblesse : la précision est de seulement 0.78, malgré un rappel parfait (1.00), ce qui signifie que beaucoup d'images ont été classées à tort comme "saines" alors qu'elles ne l'étaient pas.
- Les moyennes macro (0.91 précision, 0.96 rappel, 0.93 f1-score) et pondérée (0.95, 0.94, 0.94) indiquent des performances globalement bonnes, mais moins constantes selon les classes.
- L'accuracy globale est de 0.94, ce qui reste acceptable mais laisse apparaître des marges d'amélioration.

3.6.4 Comparaison globale entre les deux modèles

3.6.4.1 Comparaison entre DenseNet121 et UNet

Critère	DenseNet121	UNet
Accuracy	0.99	0.94
Précision (moy. pondérée)	0.99	0.95
Rappel (moy. pondérée)	0.99	0.94
F1-score (moy. pondérée)	0.99	0.94
Faiblesse principale	Aucune notable	Faible précision pour classe saine (0.78)

Tableau 3.5 : Comparaison globale des performances entre DenseNet121 et U-Net

La comparaison entre les deux modèles montre que DenseNet121 surpasse UNet dans tous les indicateurs clés : précision, rappel, f1-score et exactitude globale. Il est plus cohérent entre les différentes classes et montre une meilleure capacité de généralisation. De son côté, UNet, malgré une performance respectable, souffre d'un manque de précision dans la détection des feuilles saines, ce qui peut compromettre sa fiabilité dans une application réelle.

3.6.4.2 Analyse des graphiques de comparaison entre DenseNet121 et UNet

Les deux graphiques comparent les performances de DenseNet121 et UNet.

DenseNet121 atteint rapidement une précision d'entraînement élevée et reste stable, avec une précision de validation similaire. Cela montre un apprentissage efficace et une bonne généralisation.

UNet progresse plus lentement, avec des fluctuations dans la précision d'entraînement et une validation moins cohérente, ce qui indique un apprentissage moins stable

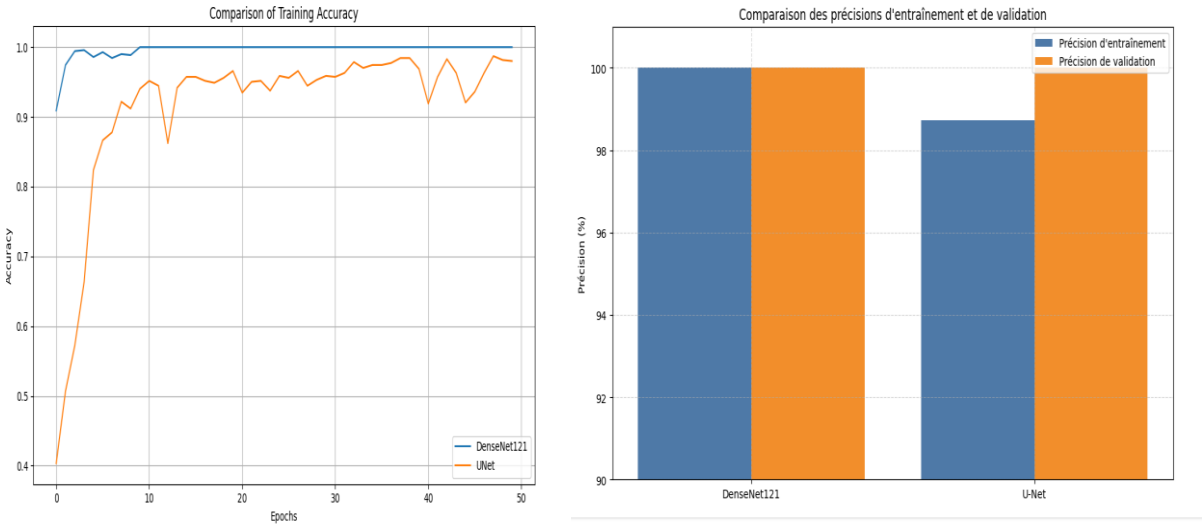


Figure 3.7 : Comparaison des précisions d'entraînement et de validation entre DenseNet121 et U-Net

3.6.5 Analyse de la Matrice de Confusion du modèle DenseNet121

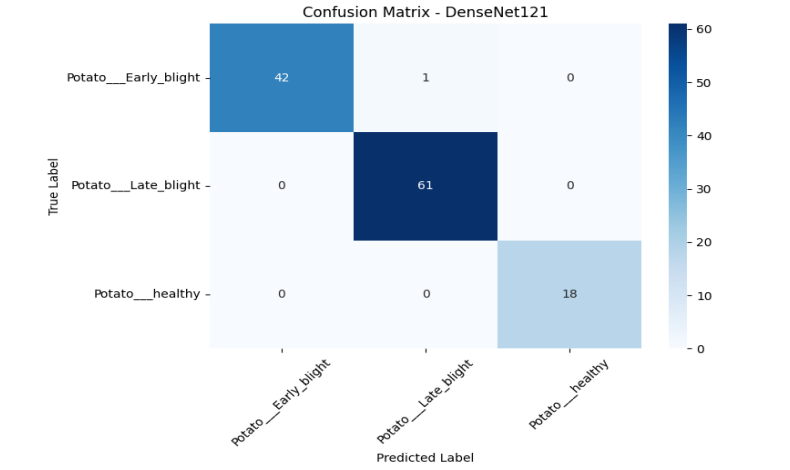


Figure 3.8 : Résultats du test du modèle Matrice de confusion

La matrice de confusion obtenue illustre la performance remarquable du modèle DenseNet121 dans la classification des feuilles de pomme de terre en trois catégories : *Potato___Early_blight*, *Potato___Late_blight* et *Potato___healthy*. Sur un total de 122 images, le modèle a correctement classé 121 cas, atteignant ainsi une précision globale de 99 %. Il a identifié avec exactitude les 61 cas de *Late_blight* et les 18 cas de feuilles saines, tandis qu'une seule erreur a été observée parmi les 43 cas de *Early_blight*, où une feuille malade a été confondue avec *Late_blight*.

Cette confusion est compréhensible compte tenu de la ressemblance visuelle entre les deux types de maladies. Ces résultats démontrent la capacité du modèle à distinguer efficacement entre feuilles saines et malades, ainsi qu'entre les différentes pathologies, ce qui confirme sa fiabilité pour un usage en diagnostic automatique des maladies de la pomme de terre.

3.7 Conclusion

Ce chapitre a permis de concrétiser l'utilisation des modèles d'apprentissage profond dans une application web fonctionnelle, destinée à la reconnaissance des maladies des feuilles de pomme de terre. L'intégration réussie des modèles DenseNet et U-Net dans une plateforme conviviale démontre le potentiel réel de l'intelligence artificielle pour répondre aux défis actuels du secteur agricole.En plus de la précision des prédictions, la rapidité du système et sa simplicité d'utilisation en font un outil prometteur pour l'aide à la décision dans le domaine de la santé des plantes. Cette démarche ouvre ainsi la voie à d'autres applications similaires dans le futur, visant à soutenir l'agriculture de précision à travers des technologies intelligentes et automatisées.

Conclusion générale

Conclusion générale

Dans ce mémoire, nous avons exploré l'utilisation des techniques d'intelligence artificielle, en particulier l'apprentissage profond à travers les réseaux de neurones convolutifs (CNN), pour la reconnaissance automatique des maladies des plantes à partir d'images. Cette approche s'inscrit dans le contexte de l'agriculture intelligente, visant à améliorer la détection précoce des maladies et à réduire les pertes agricoles liées à des diagnostics tardifs ou imprécis.

Nous avons d'abord étudié les bases théoriques de l'intelligence artificielle, de l'apprentissage automatique et des différentes architectures de réseaux neuronaux profonds. Ensuite, nous avons analysé les étapes essentielles dans la construction d'un système de classification d'images, depuis la collecte et le prétraitement des données jusqu'à la phase de modélisation et d'évaluation des performances.

Enfin, nous avons mis en œuvre un système basé sur les CNN, en particulier à l'aide de modèles puissants comme DenseNet, pour reconnaître plusieurs maladies affectant les feuilles de plantes. Les résultats obtenus ont montré que cette approche est non seulement réalisable mais aussi efficace, notamment en termes de précision et de robustesse.

Cependant, certaines limitations subsistent, telles que la dépendance à des jeux de données bien annotés et la sensibilité du modèle aux conditions de prise de vue. Ces aspects ouvrent la voie à des perspectives futures, notamment l'amélioration de la généralisation du modèle, l'intégration de données en temps réel via des applications mobiles, ou encore l'utilisation de techniques d'apprentissage par transfert pour renforcer les performances sur des bases de données limitées.

Ainsi, cette étude démontre que l'intelligence artificielle représente une solution prometteuse pour accompagner les agriculteurs et les chercheurs dans la surveillance et la gestion des maladies des plantes, contribuant à une agriculture plus durable, efficace et résiliente.

Bibliographie:

[1] Gardner, H. (1983). Frames of Mind. The Theory of Multiple Intelligences. Basic Books.

[2] Chen, R., & Chen, C. (2022). Artificial intelligence. An introduction for the inquisitive reader. CRC Publisher.

[3] Stadlmann, C., &Zehetner, A. (2021). Human Intelligence Versus Artificial Intelligence: A Comparison of Traditional and AI-Based Methods for Prospect Generation. In Marketing and Smart Technologies (pp. 11-22). Springer.

[4] De Cremer, D., &Kasparov, G. (2021). AI should augment human intelligence, not replace it. Harvard Business Review, 18.

[5] Bleakley, C. (2020). Poems That Solve Puzzles: The History and Science of Algorithms. OUP Oxford.

[6] Bartneck, C., Lütge, C., Wagner, A., & Welsh, S. (2021). What Is AI? In C. Bartneck, C. Lütge, A. Wagner, & S. Welsh (Eds.), An Introduction to Ethics in Robotics and AI (pp. 5-16).

[7] Bashir et al., 2016] Bashir, S., Qamar, U., Khan, F. H., and Naseem, L. (2016). Hmv: A medical decision support framework using multi-layer classifiers for disease prediction. Journal of Computational Science, 13:10–25

[8] Coronato et al., 2020] Coronato, A., Naeem, M., De Pietro, G., and Paragliola, G. (2020). Reinforcement learning for intelligent healthcare applications: A survey. Artificial Intelligence in Medicine.

[9] Yousef poor et al., 2021] Yousef poor, M. S., Yousef poor, E., Barati, H., Barati, A., Movaghar, A., and Hosseinzadeh, M. (2021). Secure data aggregation methods and countermeasures against various attacks in wireless sensor networks: A comprehensive review. Journal of Network and Computer Applications.

[10] [Seaton, 2021] Seaton, H. (2021). The construction technology handbook. John Wiley & Sons.

[11] Kotsiantis, S. B., Zaharakis, I., and Pintelas, P. Supervised machine learning: A review of classification techniques. Emerging artificial intelligence applications in computer engineering, 160:3–24, 2007.

[12] ImageSource: https://static.javatpoint.com/tutorial/machine-learning/images/regression-vs-classification-in-machine-learning.png (Consulté le 15/04/2025)

[13] Bengio, Y., Courville, A. C., and Vincent, P. Unsupervised feature learning and deep learning: A review and new perspectives. CoRR.

[14] Chapelle, O., Scholkopf, B., and Zien, A. Semi-supervised learning (chapelle, o. et al., eds.; 2006) [book reviews]. IEEE Transactions on Neural Networks, 20(3):542–542, 2009.

[15] Franc¸ois-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., and Pineau, J. An introduction to deep reinforcement learning. arXiv preprint arXiv:1811.12560, 2018

[16] Lahlou OUCHIHA. « Classification supervisée de documents : étude comparative ». Thèse de doct. Université du Québec en Outaouais, 2016.

[17] Hassane HILALI. « Application de la classification textuelle pour l'extraction des règles d'association maximales ». Thèse de doct. Université du Québec à Trois-Rivières, 2009.

[18] Hassan CHOUAIB. « Sélection de caractéristiques : méthodes et applications ». In: Paris Descartes University: Paris, France (2011).

[19] Qu, K. (2024). Research on linear regression algorithm. *MATEC Web of Conferences, 395*, 01046

[20] Alain GIRARD. « Exploration d'un algorithme génétique et d'un arbre de décision à des fins de catégorisation ». Thèse de doct. Université du Québec à Trois-Rivières, 2007.

[21] Mouhoub BELAZZOUG. « Apprentissage statistique pour l'extraction des relations à partir de textes ». Thèse de doct. 2021.

[22] https : / / www. Futura - sciences . com / tech / definitions / informatique - reseau-neuronal601/. (dernier accès : 05 /2025.)

[23] Nadia BENAHMED. Optimisation de réseaux de neurones pour la reconnaissance de chiffres manuscrits isolés : Sélection et pondération des primitives par algorithmes génétiques. École de technologie supérieure, 2002.

[24] Mouhoub BELAZZOUG. « Apprentissage statistique pour l'extraction des relations à partir de textes ». Thèse de doct. 2021.

[25] Pierre-Louis GONZALEZ, (2008). "MÉTHODES DE CLASSIFICATION", Cnam.

26 WikiDocs. (n.d.). *Deep Learning Bible - Artificial Neural Networks*. Retrieved February 15, 2025.

[27] Turing. (n.d.). *How neural network models in machine learning work.* Retrieved February 15, 2025.

[28] Teuwen, J., & Moriakov, N. (2020). *Convolutional neural networks. Handbook of Medical Image Computing and Computer Assisted Intervention, 481–501.*

[29]. G. Huang, Z. Liu, L. Van Der Maaten, & K.Q. Weinberger (2017) Densely Connected Convolutional Networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu,

[30] Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2021). *Dive into Deep Learning*.

[31] Najam, A. (2023). *Temporal Localization of Representations in Recurrent Neural Networks* (Master's thesis). Dalarna University

[32] Fuzhen Zhuang et al. "A comprehensive survey on Transfer Learning". In : Proceedings of the IEEE 109.1 (2021), pp. 43–76.

[33] University of Kentucky**.** (n.d.). *Plant Diseases: Identification and Management*. Retrieved from https://www.uky.edu/Ag/Entomology/PSEP/pdfs/11pests1disease.pdf (Consulté le 03/01//2025)

[34] Camargo, A., Smith, J. S. (2009). An image-processing based algorithm to automatically identify plant disease visual symptoms. *Biosyst. Eng.* 102, 9–21.

[35] Basavaiah, J., Anthony, A. A. (2020). Tomato leaf disease classification using multiple feature extraction techniques. *Wireless Pers. Commun.* 115, 633–515.

[36] SONKA, Milan; HLAVAC, Vaclav; BOYLE, Roger. Image pre-processing. In: Image Processing, Analysis and Machine Vision. Boston, MA : Springer US, 1993, p. 56-111.

[37] BERA, Tanmoy; DAS, Ankur; SIL, Jaya; DAS, Asit K. A survey on rice plant di- Sease identification using image processing and data mining techniques. In: Emerging Technologies in Data Mining and Information Security. Springer, 2019, p. 365-376.

[38] VINCENT, Guesdon. Détection efficace de contours d'images. 2004. Thèse de doct. Université du Québec en Outaouais.

[39] SINGH, Vijai; MISRA, Ak K. Detection of plant leaf diseases using image segmentation and soft computing techniques. Information processing in Agriculture. 2017, t. 4, nº 1, p. 41-49.

[40] Basavaiah, J., Anthony, A. A. (2020). Tomato leaf disease classification using multiple feature extraction techniques. *Wireless Pers. Commun.* 115, 633–515.

[41] Yao, Q., Guan, Z., Zhou, Y., Tang, J., Hu, Y., Yang, B. (2009). "Application of support vector machine for detecting rice diseases using shape and color texture features," in *2009 IEEE Int. Conf. Eng. Comput.* Hong Kong, China, 79–83.

[42] Sugimura, D., Mikami, T., Yamashita, H., Hamamoto., T. (2015). Enhancing color images of extremely low light scenes based on RGB/NIR images acquisition with different exposure times. *IEEE Trans. Image Process.* 24, 3586–3975.

[43] Huang, J., Kumar, S. R., Mitra, M., Zhu, W.-J., Zabih, R. (1997). "Image indexing using color correlograms," in *Proceedings of IEEE computer society conference on computer vision and pattern recognition*, San Juan, Puerto Rico, USA 762–768.

[44] Kaur, S., Pandey, S., Goel., S. (2019). Plants disease identification and classification through leaf images: A survey. *Arch. Comput. Methods Eng.* 26, 507–305.

[45] Mokhtar, U., Bendary, N. E., Hassenian, A. E., Emary, E., Mahmoud, M. A., Hefny, H., et al. (2015). "SVM-Based Detection of Tomato Leaves Diseases," in *Intelligent Systems'2014* (Springer, Cham), Warsaw, Poland, 641–652.

[46] Islam, M., Dinh, A., Wahid, K., Bhowmik, P. (2017). "Detection of potato diseases using image segmentation and multiclass support vector machine," in 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), Windsor, ON, Canada. 1–4.

[47] Kumar R, Chug A, Singh AP, Singh D. A systematic analysis of machine learning and deep learning-based approaches for plant leaf disease classification: a Review. J Sensors. 2022.

[48] Saleem MH, Potgieter J, Arif KM. Plant disease classification: a comparative evaluation of convolutional neural networks and deep learning optimizers. Plants. 2020 ;9(10) :1–17.

[49] Tiwari V, Joshi RC, Dutta MK. Dense convolutional neural networks based multiclass plant disease detection and classification using leaf images. Ecole Inform. 2021;63: 101289.

[50] KAWASAKI, Yusuke; UGA, Hiroyuki; KAGIWADA, Satoshi; IYATOMI, Hitoshi. Basic study of automated diagnosis of viral plant diseases using convolutional neural networks. In International symposium on visual computing. 2015, p. 638-645.

[51] FUJITA, Erika; KAWASAKI, Yusuke; UGA, Hiroyuki; KAGIWADA, Satoshi; IYATOMI, Hitoshi. Basic investigation on a robust and practical plant diagnose- : tic system. In 2016 15th IEEE international conference on machine learning and applications (ICMLA). 2016, p. 989-992.

[52] SLADOJEVIC, Srdjan; ARSENOVIC, Marko; ANDERLA, Andras; CULIBRK, Dubravko; STEFANOVIC, Darko. Deep neural networks-based recognition of plant diseases by leaf image classification. Computational intelligence and neuroscience. 2016, t. 2016.

[53] MOHANTY, Sharada P; HUGHES, David P; SALATHÉ. Marcel. Using deep lear- ning for image-based plant disease detection. Frontiers in plant science. 2016, t. 7, p. 1419.

[54] AMARA, Jihen; BOUAZIZ, Bassem; ALGERGAWY, Alsayed. A deep learning- based approach for banana leaf diseases classification. Daten bank system für Busi- ness, Technologie und Web (BTW 2017)-Workshop band. 2017.

[55] LIU, Weibo; WANG, Zidong; LIU, Xiaohui; ZENG, Nianyin; LIU, Yurong; AL- SAADI, Fuad E. A survey of deep neural network architectures and their applications. Neurocomputing. 2017, t. 234, p. 11-26.

[56] LOEY, Mohamed; ELSAWY, Ahmed; AFIFY, Mohamed. Deep learning in plant diseases detection for agricultural crops: a survey. International Journal of Service Science, Management, Engineering, and Technology (IJSSMET). 2020, t. 11, nº 2, p. 41-58.

[57] TOO, Edna Chebet; YUJIAN, Li; NJUKI, Sam; YINGCHUN, Liu. A comparative study of fine-tuning deep learning models for plant disease identification. Computers and Electronics in Agriculture. 2019, t. 161, p. 272-279.

[58] GANDHI, Rutu; NIMBALKAR, Shubham; YELAMANCHILI, Nandita ; PONKSHE, Surabhi. Plant disease detection using CNNs and GANs as an augmentative approach. In 2018 IEEE International Conference on Innovative Research and Development (ICIRD). 2018, p. 1-5.

[54] AMARA, Jihen ; BOUAZIZ, Bassem ; ALGERGAWY, Alsayed, "A deep learning-based approach for banana leaf diseases classification", *Datenbank-Systeme für Business, Technologie und Web (BTW 2017) – Workshopband*, 2017.

[55] LIU, Weibo ; WANG, Zidong ; LIU, Xiaohui ; ZENG, Nianyin ; LIU, Yurong ; ALSAADI, Fuad E., "A survey of deep neural network architectures and their applications", *Neurocomputing*, **234** (2017) 11-26.

[56] LOEY, Mohamed ; ELSAWY, Ahmed ; AFIFY, Mohamed, "Deep learning in plant diseases detection for agricultural crops: a survey", *International Journal of Service Science, Management, Engineering, and Technology (IJSSMET)*, **11**, nº 2 (2020) 41-58.

[57] TOO, Edna Chebet ; YUJIAN, Li ; NJUKI, Sam ; YINGCHUN, Liu, "A comparative study of fine-tuning deep learning models for plant disease identification", *Computers and Electronics in Agriculture*, **161** (2019) 272-279.

[58] GANDHI, Rutu ; NIMBALKAR, Shubham ; YELAMANCHILI, Nandita ; PONKSHE, Surabhi, "Plant disease detection using CNNs and GANs as an augmentative approach", *2018 IEEE International Conference on Innovative Research and Development (ICIRD)*, 2018, p. 1-5.

[59] RONNEBERGER, O., et al. “U-Net: Convolutional Networks for Biomedical Image Segmentation”,MICCAI,2015.

[60] MOHANTY, S.P., et al.”Using Deep Learning for Image-Based Plant Disease Detection. Plant Phenomics”, 2016.

[61] FUENTES, A., et al. “Deep Learning for Segmentation of Plant Diseases”, Computers and Electronics in Agriculture,2021.

[62] HUANG, G., et al.”Densely Connected Convolutional Networks”,CVPR, 2017

[63] ZHANG, X., et al.”Comparative Analysis of Deep Learning Models for Crop Disease Recognition”, IEEE Access,2022.

Annexes

Environnement de travail et outils de développement

Plateformes utilisées

- **Google Colab** : Plateforme basée sur Jupyter Notebook offrant un accès gratuit à des GPU (ex. : NVIDIA Tesla T4), utilisée pour l'entraînement du modèle CNN.
- **Google Drive** : Utilisé comme espace de stockage centralisé pour les datasets, les modèles sauvegardés (.h5,. pb) et les résultats.
- **Kaggle** : Source principale du jeu de données contenant des images de feuilles de plantes malades.

Environnements de développement (IDE)

- **PyCharm** : Utilisé pour le développement du back-end avec FastAPI, grâce à ses outils de débogage et de gestion d'environnement virtuel.
- **Visual Studio Code (VS Code)** : Utilisé pour le développement du front-end (ReactJS) et pour le scripting Python léger.
- **Jupyter Notebook** : Utilisé dans Google Colab pour exécuter et tester le code Python de manière interactive.

Outils et bibliothèques

- **Python (v3.9)** : Langage principal du projet, reconnu pour sa flexibilité en intelligence artificielle.
- **TensorFlow & Keras** : Frameworks de deep learning utilisés pour construire et entraîner les modèles CNN.
- **NumPy** : Utilisé pour la manipulation de tableaux numériques et les opérations mathématiques.
- **Matplotlib** : Employé pour la visualisation des résultats (précision, courbes de perte, etc.).
- **FastAPI** : Framework utilisé pour créer une API REST performante.

- **Uvicorn** : Serveur ASGI léger utilisé pour héberger l'API FastAPI.
- **ReactJS** : Bibliothèque JavaScript pour construire une interface utilisateur réactive.
- **Node.js & NPM** : Utilisés pour gérer les dépendances JavaScript et exécuter les composants React.

Ressources matérielles

- **Plateforme d'entraînement** : Google Colab, avec GPU NVIDIA Tesla T4 (2560 cœurs CUDA, 16 Go de RAM GPU).
- **Ordinateur local** :
  - **Processeur** : Intel Core i5-4210M @ 2.60GHz
  - **RAM** : 8 Go
  - **Carte graphique** : Intel HD Graphics 4600
  - **Stockage** : 224 Go SSD

Développement

Dans cette section, nous allons nous concentrer sur les éléments essentiels à la construction d'une application web. Ces composantes fondamentales sont généralement présentes dans tout projet de développement, qu'il s'agisse d'applications web, mobiles, de jeux ou d'autres types de logiciels.

Ce travail met principalement l'accent sur les aspects techniques de la création d'une application web monopage (SPA).

Il est important de souligner que d'autres aspects, tels que la sécurité des applications, bien qu'importants, ne seront pas traités en profondeur ici. Néanmoins, tout développeur doit les prendre en compte lors de la mise en œuvre des fonctionnalités côté backend et frontend. La sécurité constitue en effet un pilier de l'infrastructure et de la surveillance d'une application.

Le développement de notre application web pour le diagnostic des maladies des plantes se divise en deux parties principales :

Côté serveur (Backend)

Le backend représente la partie serveur de l'application web. Il héberge le modèle d'apprentissage profond développé précédemment, ainsi que les noms des classes et les prétraitements appliqués aux images. Le backend veille également au bon fonctionnement de l'ensemble du système du côté client.

Cette partie du site n'est ni visible ni directement accessible à l'utilisateur final. Toutefois, elle est indirectement sollicitée à travers les requêtes envoyées depuis l'interface frontend.

Parmi les tâches courantes assurées par le backend, on retrouve l'écriture d'API, la création de bibliothèques, et l'interaction avec des composants système sans interface graphique. Ces tâches peuvent également concerner des systèmes de calculs scientifiques.

Choix de la technologie

Pour développer le backend de notre application, nous avons choisi la bibliothèque FastAPI de Python, comme mentionné précédemment. Le processus passe par plusieurs étapes décrites ci-dessous :

- **Installation des prérequis**

Avant de commencer à coder, il est nécessaire d'installer FastAPI et les bibliothèques associées. Nous avons opté pour un environnement virtuel, permettant de gérer proprement toutes les dépendances nécessaires au développement et au déploiement.

- **Installation de FastAPI et Uvicorn**

FastAPI ne dispose pas de serveur de développement intégré. Nous utilisons donc Uvicorn, un serveur ASGI (Asynchronous Server Gateway Interface), pour exécuter notre application.

```
Pip install fastapi uvicorn
```

Ensuite, nous créons un répertoire qui contiendra tous les fichiers nécessaires au développement backend, notamment le fichier principal main.py, qui orchestre l'ensemble des fonctionnalités de l'application.

- **Exécution de l'API**

Le fichier main.py contient toutes les routes et opérations. Pour le lancer, on ouvre un terminal dans le répertoire du projet, puis on exécute la commande suivante :

```
Uvicorn main : app –reload
```

Une fois cette commande exécutée, l'application backend démarre et devient accessible localement.

- **Importation des bibliothèques**

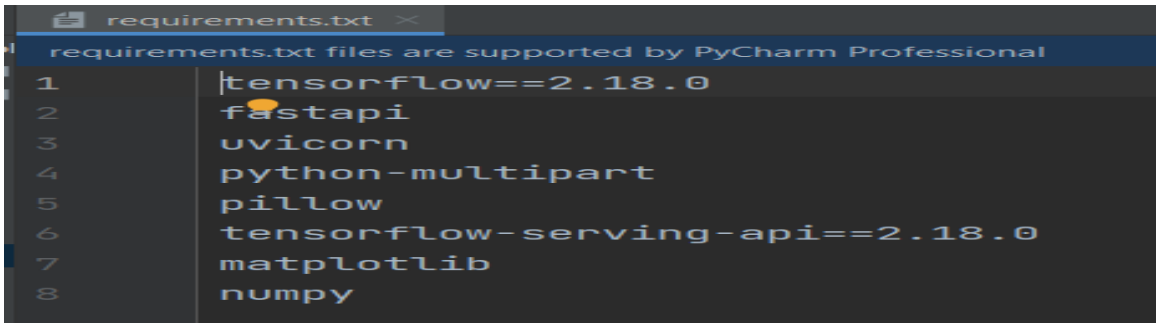
Afin d'assurer le bon fonctionnement du programme, il est impératif d'importer toutes les bibliothèques utilisées pour entraîner notre modèle CNN.

Pour cela, nous générons un fichier requirements.txt à l'aide des commandes suivantes :

```
pip install pipreqs

pipreqs Web_application/api
```

Après exécution, le fichier requirements.txt regroupe toutes les bibliothèques nécessaires à notre projet.



On installe les librairies en utilisant la commande :

```
Pip install -r requirements.txt
```

Maintenant que notre environnement de développement est prêt, programmation du côté serveur. On aborde la partie

- **Prétraitement des images**

Avant d'effectuer la détection proprement dite, les images chargées dans l'application doivent subir un prétraitement. Ce processus se déroule comme suit :

1. Conversion de l'image en tableau Numpy via numpy. Array ().
2. 2.Ajout d'une dimension supplémentaire avec numpy.expand_dims() pour correspondre au format d'entrée du modèle.
3. Redimensionnement des valeurs dans l'intervalle [0, 1] (à partir de [0, 255]) afin de normaliser les images et assurer une cohérence avec l'entraînement du modèle CNN.

```
def read_file_as_image(data) -> np.ndarray:
    image = np.array(Image.open(BytesIO(data)))
    return image

@app.post("/predict")
async def predict(
    file: UploadFile = File(...)
):
    image = read_file_as_image(await file.read())
    img_batch = np.expand_dims(image, 0)

    predictions = MODEL.predict(img_batch)

    predicted_class = CLASS_NAMES[np.argmax(predictions[0])]
    confidence = np.max(predictions[0])
    return {
```

- **Diagnostic**

C'est à ce niveau qu'intervient l'étape la plus importante de notre backend, il s'agit de la partie du code responsable du diagnostic des différentes maladies.

Lorsque l'utilisateur charge une image d'une feuille de plante, l'application web doit être capable de reconnaître le type de plante concernée ainsi que la maladie associée à ce type.

Pour ce faire, nous utilisons le fichier modèle (model file) que nous avons préalablement enregistré dans le même répertoire du projet. Le chargement de ce modèle s'effectue à l'aide de la fonction load_model, comme indiqué ci-dessous :

```
MODEL = tf.keras.models.load_model("../models/1.keras")
```

Nous avons maintenant un autre élément essentiel : @app. Route('/predict').

Cet endroit dans le code associe la fonction predict () à l'URL /predict.

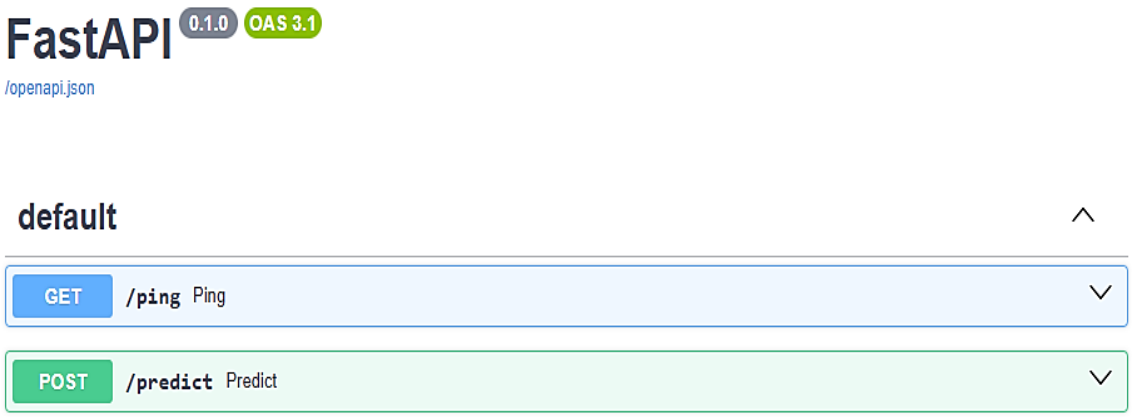
Comme son nom l'indique, cette URL est appelée lorsque l'utilisateur soumet une image. Elle permet de :

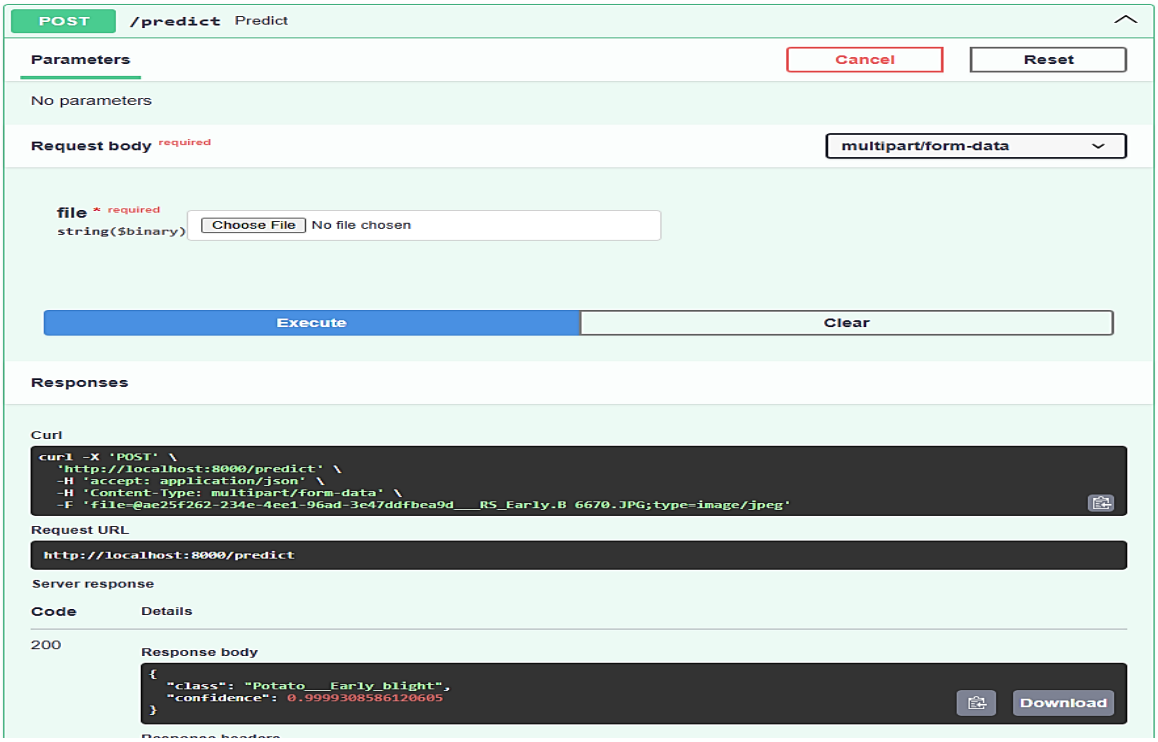
- Recevoir l'image transmise par l'utilisateur,
- Appliquer les pré-traitements nécessaires à cette image,
- Puis la faire passer à travers les différentes couches du modèle CNN que nous avons entraîné.

Le modèle renvoie ensuite un score de confiance pour chaque classe possible, en utilisant la fonction softmax().

Ensuite, la fonction argmax () est utilisée pour déterminer la classe ayant le score le plus élevé. Ce score correspond à l'indice d'une classe dans le tableau class_names, qui contient la liste de toutes les classes d'aliments ou maladies possibles.

Enfin, pour exprimer le score de confiance sous forme de pourcentage, nous le multiplions par 100, ce qui permet d'obtenir une plage de valeurs comprise entre 1 et 100. Les figures ci-dessous illustrent l'interface de notre serveur backend.





Côté utilisateur (Frontend)

La partie d'un site web avec laquelle l'utilisateur interagit directement est appelée le frontend, également connu sous le nom de côté client de l'application. Elle englobe tout ce que l'utilisateur visualise dans le navigateur : les couleurs, les styles de texte, les images, les graphiques, les tableaux, les boutons ainsi que le menu de navigation. Le développement du frontend repose sur l'utilisation des langages HTML, CSS et JavaScript.

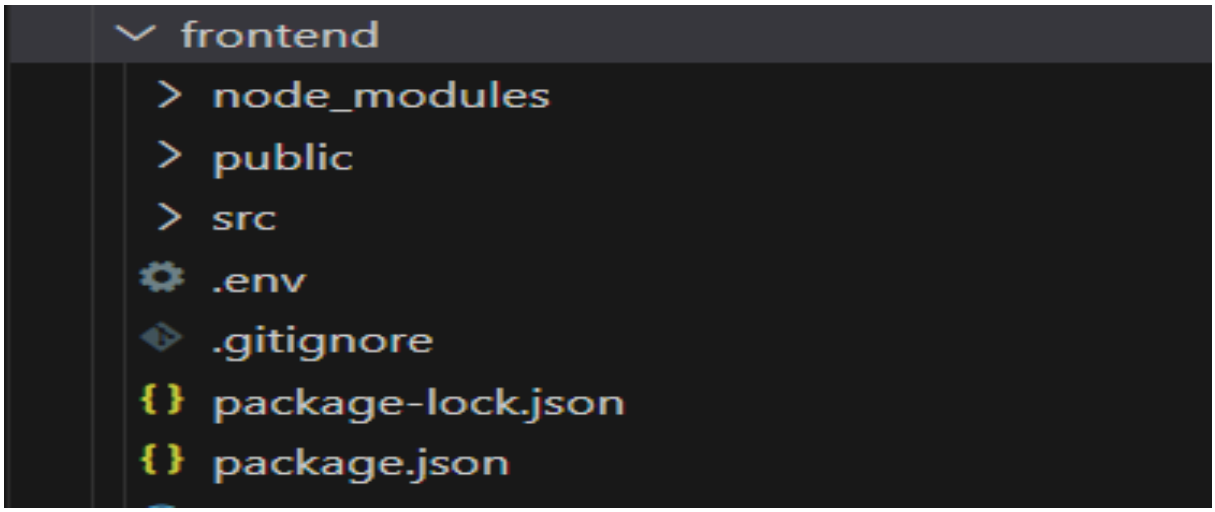
Les développeurs frontend sont chargés de la structure, de la conception, du comportement et du contenu visible sur l'écran lorsqu'un site web, une application web ou mobile est ouvert. Deux objectifs essentiels du frontend sont la réactivité et la performance.

Il est indispensable de veiller à ce que le site soit réactif, c'est-à-dire que l'interface homme-machine soit à la fois ergonomique et efficace. Ces interfaces doivent également être simples d'utilisation et faciles à comprendre pour les utilisateurs.

Comme mentionné précédemment, nous utiliserons la bibliothèque React JS, ainsi que HTML et CSS pour concevoir notre interface web. Avant de commencer la phase de développement, il est nécessaire d'installer le logiciel Node.js.

Répertoires

L'application React crée automatiquement les dossiers requis, comme indiqué ci-dessous.



Structure du projet React

- **Répertoire node_modules**

Ce dossier contient toutes les dépendances et sous-dépendances nécessaires au fonctionnement du projet. Bien que nous n'ayons ajouté que React et React Scripts, ces derniers dépendent de nombreuses autres bibliothèques, toutes regroupées dans ce répertoire.

- **Répertoire public**

Ce dossier contient principalement trois fichiers :

  - **cblogo.png :** une icône affichée dans l'onglet du navigateur.
  - **index.html** : fichier HTML principal. Il représente le point d'entrée de l'application dans le navigateur. Le répertoire public, également appelé dossier racine, est celui qui est servi par le serveur web. Ce fichier HTML unique est fondamental dans tout projet React.

- **Répertoire src**

Ce dossier contient le code source de l'application React. C'est ici que le développement principal a lieu. On peut y ajouter ses propres sous-dossiers. Pour optimiser les performances, seuls les fichiers présents dans ce dossier sont traités par Webpack. Par défaut, ce répertoire comprend les fichiers suivants :

- ✓ **Home.js :** fichier principal de l'interface utilisateur. Il définit le design global de l'application (taille, couleurs, position des éléments, etc.).
- ✓ **App.css :** contient des classes CSS qui définissent les styles utilisés dans le fichier App.js.
- ✓ **App.js :** un composant React de base appelé App, fourni automatiquement lors de la création d'une nouvelle application. Ce composant est rendu par le fichier index.js. Toute modification dans ce fichier se reflétera en temps réel sur l'adresse http://localhost:3000.
- ✓ **App.test.js :** permet de créer des tests unitaires pour vérifier le bon fonctionnement de divers composants.
- ✓ **index.css :** fichier contenant le style de base de l'application.
- ✓ **index.js :** point d'entrée principal. Ce fichier est exécuté lorsque l'on lance le projet.

- **Fichier. Gitignore :**

Utilisé par les outils de gestion de version, ce fichier indique quels fichiers ou dossiers doivent être ignorés (comme node_modules) lors de la synchronisation du code avec un dépôt Git.

- **Fichier package. Json :**

Fichier standard dans tout projet Node.js. Il contient des informations essentielles comme le nom du projet, sa version, les scripts de lancement, et surtout, la liste des dépendances. Lorsqu'une nouvelle bibliothèque est installée, elle est automatiquement enregistrée ici.

- **Fichier. env**

Ce fichier permet de stocker des variables d'environnement, telles qu'une clé d'API ou l'URL du serveur backend. Cela facilite la communication entre le frontend et le backend. Dans notre cas, il contient la ligne suivante :

```
REACT_APP_API_URL=http://localhost:8000/predict
```

Cette variable permet de rediriger les requêtes du frontend vers le backend, exécuté sur le port 8000.