REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de L'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed Khider – BISKRA

Faculté des Sciences Exactes

Département d'informatique

N°d'ordre: IVA01/M2/2025

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours: Images et Vie Artificielle(IVA)

Réalisation d'une application pour la détection des maladies du blé

Réalisé Par :

Bey Meriem & Rhalmi Imane

Encadré Par :

Zerari Abd El Mouméne

Année universitaire 2024-2025

Remerciements

C’est avec beaucoup de fierté et d’humilité que nous exprimons notre plus profonde gratitude à Allah le Tout-Puissant, dont la grâce et la miséricorde ont permis cette importante réalisation. Son soutien indéfectible a été la lumière qui a guidé notre chemin vers le succès.

Nous exprimons également notre sincère reconnaissance à notre estimé superviseur, ***Dr. Zerari Abd El Mouméne****, qui a généreusement partagé ses vastes connaissances et nous a fourni des conseils inestimables tout au long de la préparation de ce travail. Ses idées éclairées et sa direction ont eu un impact significatif sur notre développement académique et nous ont permis d'acquérir des compétences essentielles pour notre avenir.*

Nous remercions chaleureusement les membres du jury pour avoir accepté d’évaluer notre mémoire. Leur retour et leur considération nous encouragent à poursuivre notre parcours académique avec encore plus de détermination et de persévérance.

Nous adressons une pensée toute particulière à nos parents, pour leur amour, leurs prières, leur patience et leur soutien indéfectible. Leur confiance en nous a été une source constante de motivation et de force.

Enfin, nous remercions du fond du cœur toutes les personnes qui, de près ou de loin, ont contribué à ce voyage : amis, collègues, camarades et proches. Leur soutien, leurs encouragements et leur présence bienveillante ont été le carburant de notre persévérance.

À vous tous, nous disons, du plus profond de notre cœur : Merci.

Imane & Meriem

Résumé

À l'heure où la transformation numérique révolutionne tous les secteurs, ce projet se positionne comme une initiative pionnière pour optimiser la production céréalière. Il s'agit de développer une application intuitive capable de réinventer la manière dont les maladies du blé sont détectées et gérées par les agriculteurs.

Le cœur de l'application réside dans sa capacité à analyser des images de feuilles de blé capturées via un simple smartphone. Bien plus qu'un outil de reconnaissance visuelle, elle utilise l'intelligence artificielle pour établir un diagnostic fiable et précis, en tenant compte des variations d'éclairage et des conditions environnementales.

Une analyse comparative des approches existantes nous a permis de concevoir une architecture complète couvrant l'ensemble du pipeline : de la collecte et préparation des données à l'entraînement, l'évaluation et l'intégration du modèle dans une application mobile fonctionnelle. Le modèle ViT, au cœur de notre solution, a atteint un taux de précision de 98 %, surpassant d'autres architectures de référence telles que ResNet50V2 et MobileNetV2, ce qui confirme sa capacité à généraliser efficacement à de nouveaux cas.

Après apprentissage et évaluation, le modèle est déployé en environnement réel et intégré à une application web via le framework Gradio.

L'application développée se distingue non seulement par sa performance, mais aussi par son interface simple et intuitive, la rendant particulièrement utile pour les agriculteurs, techniciens et chercheurs. Elle permet une détection précoce des maladies, favorisant ainsi une gestion phytosanitaire plus ciblée et une réduction de l'usage excessif de produits chimiques.

Mots-clés : Blé, maladies, détection précoce, application mobile, agriculture intelligente, intelligence artificielle, durabilité, rendement.

Abstract

At a time when digital transformation is revolutionizing every sector, this project stands out as a pioneering initiative aimed at optimizing cereal production. It focuses on developing an intuitive application capable of reinventing how wheat diseases are detected and managed by farmers.

The core of the application lies in its ability to analyze images of wheat leaves captured using a simple smartphone. More than just a visual recognition tool, it leverages artificial intelligence to provide a reliable and accurate diagnosis, taking into account lighting variations and environmental conditions.

A comparative analysis of existing approaches enabled us to design a complete architecture covering the entire pipeline—from data collection and preprocessing to model training, evaluation, and integration into a functional mobile application. The Vision Transformer (ViT) model, at the heart of our solution, achieved a 98% accuracy rate, outperforming other reference architectures such as ResNet50V2 and MobileNetV2, thus confirming its strong generalization capabilities to new cases.

After training and evaluation, the model was deployed in a real-world environment and integrated into a web application using the Gradio framework.

The developed application stands out not only for its technical performance but also for its simple and intuitive interface, making it particularly useful for farmers, technicians, and researchers. It enables early disease detection, thus promoting more targeted phytosanitary management and reducing the excessive use of chemical products.

Keywords : Wheat, diseases, early detection, mobile application, smart agriculture, artificial intelligence, sustainability, yield.

ملخص

في وقت تُحدث فيه التحول الرقمي ثورة في جميع القطاعات، يُعد هذا المشروع مبادرة رائدة تهدف إلى تحسين إنتاج الحبوب. يتمثل الهدف في تطوير تطبيق ذكي وسهل الاستخدام يعيد ابتكار الطريقة التي يتم بها اكتشاف أمراض القمح وإدارتها من قبل المزارعين.

يعتمد جوهر هذا التطبيق على قدرته على تحليل صور لأوراق القمح يتم التقاطها باستخدام هاتف ذكي بسيط. فهو ليس مجرد أداة للتعرف البصري، بل يستخدم تقنيات الذكاء الاصطناعي لتقديم تشخيص دقيق وموثوق، مع الأخذ في الاعتبار اختلافات الإضاءة والظروف البيئية.

سمحت لنا دراسة مقارنة للأساليب الحالية بتصميم بنية متكاملة تغطي جميع مراحل النظام: من جمع البيانات ومعالجتها، إلى تدريب النموذج وتقييمه ودمجه في تطبيق جوّال عملي. وقد حقق نموذج(Vision Transformer (ViT ، الذي يمثل محور الحل المقترح، دقة بلغت **98%**، متفوقًا على نماذج مرجعية أخرى مثل ResNet50V2 وMobileNetV2، مما يؤكد قدرته العالية على التعميم والتكيّف مع حالات جديدة.

بعد عملية التدريب والتقييم، تم نشر النموذج في بيئة حقيقية وتم دمجه ضمن تطبيق ويب باستخدام إطار العمل.Gradio

يتميّز التطبيق المطوّر ليس فقط بأدائه العالي، بل أيضًا بواجهته البسيطة والسلسة، مما يجعله مفيدًا للمزارعين والفنيين والباحثين على حد سواء. فهو يتيح الكشف المبكر عن الأمراض، مما يُسهم في تحسين إدارة الصحة النباتية والحد من الاستخدام المفرط للمواد الكيميائية.

الكلماتالرئيسية :قمح، أمراض، كشف مبكر، تطبيق جوال، زراعة ذكية، ذكاء اصطناعي، استدامة، غلة.

Table des matières

Liste des figures

Liste des tableaux

Introduction Générale

Face aux enjeux majeurs de la sécurité alimentaire mondiale et de l'évolution climatique, l'innovation technologique se révèle être un atout précieux pour l'agriculture moderne. Au premier rang de ces avancées, les applications mobiles de diagnostic phytosanitaire, dopées à l'intelligence artificielle, révolutionnent le suivi et la protection des cultures de blé. Ces outils numériques créent un lien direct entre l'expertise agronomique et les préoccupations quotidiennes des agriculteurs, offrant une détection rapide et fiable des maladies.

Ce qui rend ces technologies particulièrement attrayantes, c'est leur capacité à fournir des solutions concrètes et accessibles pour améliorer la santé des cultures. Elles s'appuient sur des algorithmes complexes, le ***deeplearning***, et la puissance de calcul des ***smartphones*** pour analyser en temps réel les symptômes révélateurs des maladies. Cette approche novatrice ouvre la voie à une intervention précoce et ciblée, réduisant ainsi les pertes de récoltes et l'utilisation excessive de produits phytosanitaires.

Dans ce contexte, la nécessité d'une identification rapide et précise des maladies du blé s'impose comme un impératif pour assurer des rendements stables et une production alimentaire durable. Que ce soit pour les petites exploitations familiales ou les vastes entreprises agricoles, la capacité à diagnostiquer promptement les maladies permet de prendre des décisions éclairées et d'optimiser les stratégies de traitement. Les applications mobiles se présentent ainsi comme une réponse pragmatique aux défis de la surveillance des cultures, offrant une expertise agronomique à portée de main.

Ce projet a pour ambition de concevoir et de réaliser une application mobile spécifiquement dédiée à la détection et à l'identification des maladies du blé. L'utilisateur pourra diagnostiquer la présence de maladies en prenant simplement un cliché de la plante, offrant ainsi une solution moderne et accessible pour répondre aux besoins grandissants de la protection des cultures.

Pour atteindre cet objectif, nous avons organisé notre projet en quatre chapitres comme suit :

Dans le premier chapitre, nous définirons les types des maladies du blé et les méthodes de diagnostique.

Chapitre1

Préliminaires et concepts de base

1.1 Introduction

Ce chapitre vise à introduire les concepts de base nécessaires à la compréhension du fonctionnement de notre application, qui se concentre sur la détection des maladies du blé.

Dans la première section, nous passerons en revue les différentes maladies du blé, leurs types et leurs symptômes, en mettant l'accent sur les méthodes de diagnostic traditionnellement utilisées.

Dans la deuxième section, nous expliquerons les concepts de base dans le domaine de l'intelligence artificielle et de l'apprentissage profond, qui constituent le pilier principal du travail de notre application, tout en mettant en évidence les techniques de reconnaissance et d'analyse d'images.

1.2 Maladies du blé

Le blé est considéré comme l'une des cultures agricoles les plus importantes au monde, mais il est sensible à de nombreuses maladies fongiques, bactériennes et virales qui affectent négativement sa productivité et la qualité de ses grains. La propagation de ces maladies entraîne des pertes économiques importantes pour les agriculteurs et menace la sécurité alimentaire.

1.2.1 Types de maladies du blé

Les maladies du blé peuvent être classées en plusieurs types principaux, en fonction du pathogène :

1.2.1.1 Rouille

Les rouilles du blé sont des maladies causées par des champignons du genre Puccinia. Elles se manifestent par l'apparition de pustules de différentes couleurs (***jaune, brune ou noire***) sur les feuilles, les tiges et parfois les épis. Les dégâts causés par les rouilles peuvent être importants, allant de la réduction du rendement à la destruction complète de la culture **[1].**

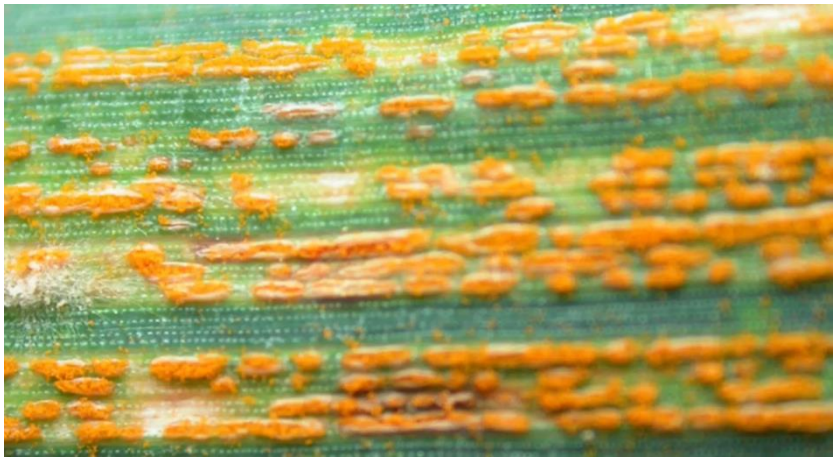

Figure 1.1 : Maladie de rouille **[1]**

1.2.1.2 Oïdium

L'oïdium du blé, causé par le champignon ***Blumeriagraminis***, est une maladie courante qui affecte les feuilles, les tiges et les épis. Il se manifeste par des pustules blanches poudreuses qui peuvent devenir grises ou brunes avec le temps, et se propage grâce aux spores transportées par le vent **[1].**

Figure 1.2 : Maladie oïdium **[1]**

1.2.1.3 Septoriose

La ***Septoriose*** du blé est une maladie causée par deux champignons différents, ***Septoriatritici*** et ***Septorianodorum***. Bien que les deux formes de ***septoriose*** puissent affecter les feuilles, elles se distinguent par leurs symptômes et leur impact sur la culture. La ***septoriose*** causée par ***Septoriatritici*** se manifeste par des taches foliaires irrégulières,

souvent plus larges que celles causées par ***Septorianodorum***. Ces taches peuvent confluer et entraîner le dessèchement de la feuille **[1].**

Figure 1.3 : Maladie Septoriose. **[1]**

1.2.1.4 Fusariose

La ***fusariose***, causée par diverses espèces de champignons du genre ***Fusarium***et ***Microdochium nivale***, est une maladie complexe qui affecte plusieurs céréales, dont le blé. Elle se manifeste par divers symptômes, allant de la fonte des semis à des lésions brunes sur les tiges et au pourridié. Un symptôme caractéristique est le blanchiment de l'épi, dû à une infection précoce lors de la floraison **[1].**

Figure 1.4 : Maladie de fusariose**[1]**

1.2.1.5 Piétin-verse

Le ***piétin-verse***, causé par les champignons ***Oculimaculayallundae*** et ***Oculimaculaacuformis***, est une maladie qui affecte principalement le blé, mais peut aussi toucher d'autres céréales. Cette maladie se manifeste par des lésions en forme d'œil à la base de la tige, avec une bordure foncée et une "***pupille***" noire au centre. Ces lésions se développent généralement sous le premier nœud et peuvent entraîner la verse de la culture **[1].**

Figure 1.5 : maladie piétin-verse **[1]**

1.2.2 Méthodes de diagnostic des maladies du blé

Il existe différentes méthodes de diagnostic des maladies du blé, et elles peuvent être classées en deux grandes catégories : les méthodes traditionnelles et les méthodes modernes soutenues par des technologies modernes.

1.2.2.1 Méthodes traditionnelles

- **Inspection visuelle**

Le diagnostic pathologique des maladies des plantes, notamment des céréales, peut être réalisé par une observation minutieuse, soit à l'œil nu, soit à l'aide d'outils simples comme une loupe à main. Il débute par une analyse du contexte, où l'historique du champ (***type de culture, traitements, stade de croissance, etc***.) est pris en compte pour écarter certaines possibilités. Ensuite, l'observation du champ permet d'examiner la répartition des

symptômes (***en lignes, foyers ou de manière uniforme***) afin de comprendre le mode de propagation de la maladie. Enfin, une observation détaillée de la plante est essentielle : il faut d'abord avoir une vue d'ensemble, puis se concentrer sur les symptômes spécifiques des feuilles et des racines, en passant des éléments macroscopiques aux détails les plus fins, pour identifier précisément la maladie **[2].**

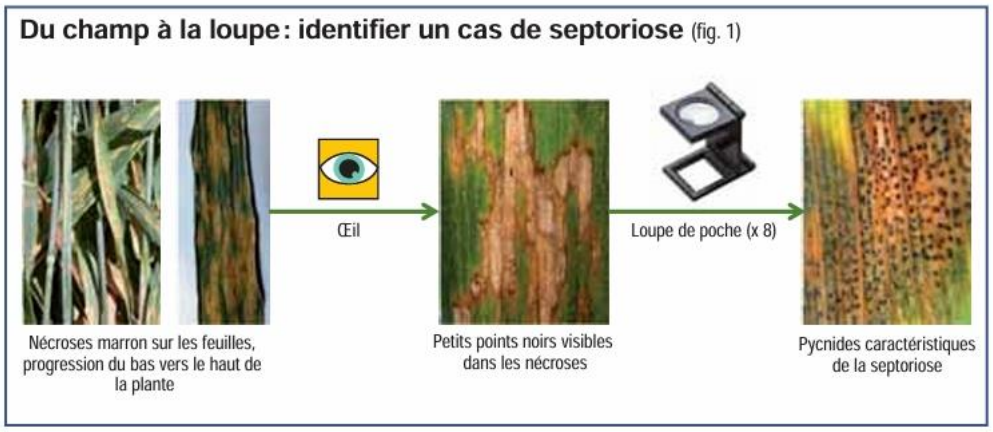

Figure 1.6 : Inspection visuelle **[3]**

- **Test de laboratoire**

L'utilisation de test de laboratoire pour évaluer l'effet de plusieurs antagonistes sur le développement de pathogènes tels que ***Fusariumsp***. et***Pyrenophoratritici-repentis***. Ces tests incluent des techniques de confrontation directe in vitro **[4].**

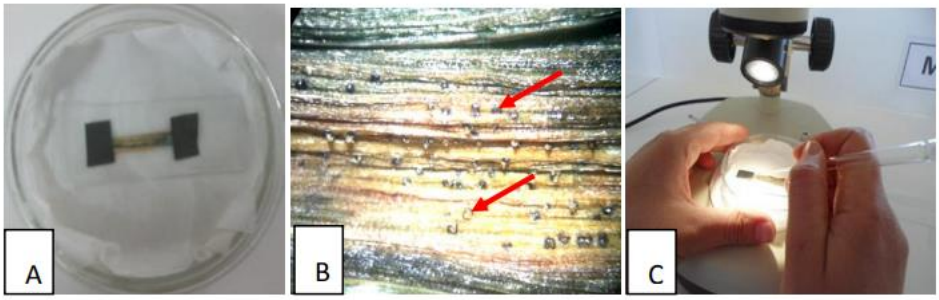

Figure 1.7 : Test laboratoire **[4]**

1.2.2.2 Méthodes modernes soutenues par la technologie

Elle s'appuie sur l'utilisation de technologies modernes pour accélérer le processus de diagnostic et augmenter sa précision. Les plus importantes de ces méthodes sont :

- **Imagerie spectrale**

La capture de données de réflectance de la lumière par la plante dans différentes longueurs d'onde, notamment dans le spectre visible et le proche infrarouge, permet d'obtenir des informations précieuses sur son état. Grâce à l'utilisation de capteurs ***multispectraux***, il est possible d'analyser en détail le spectre lumineux réfléchi par la plante. Ces données spectrales révèlent des indices sur sa composition chimique et son état physiologique, notamment les effets des maladies, en fournissant une analyse précise et non invasive de son niveau de stress ou de sa santé **[5].**

Figure 1.8 : Imagerie spectrale **[6]**

- **Intelligence artificielle**

Les techniques basées sur l'intelligence artificielle ont récemment démontré des résultats prometteurs dans la détection et la classification des maladies du blé à partir de photographies. En conséquence, ils seront mieux à même de prévenir les pertes de récoltes et d'augmenter la productivité agricole. La méthode proposée peut être utilisée pour d'autres cultures et maladies, ce qui en fait une avancée significative dans les domaines de l'agriculture et de la vision par ordinateur **[7].**

Figure 1.9 : Intelligence artificielle **[8]**

1.3. L'intelligence artificielle

L'intelligence artificielle (***IA***) est un processus d'imitation de l'intelligence humaine qui repose sur la création et l'application d'algorithmes exécutés dans un environnement informatique dynamique. Son but est de permettre à des ordinateurs de penser et d'agir comme des êtres humains.

Pour y parvenir, trois composants sont nécessaires :

- Des systèmes informatiques
- Des données avec des systèmes de gestion
- Des algorithmes d'IA avancés **[9].**

1.3.1 Apprentissage automatique

Le ***Machine Learning*** ou ***apprentissage automatique*** est un domaine scientifique, et plus particulièrement ***une sous-catégorie de l'intelligence artificielle***.

Elle consiste à laisser des algorithmes découvrir des « ***patterns*** », à savoir des motifs récurrents, dans les ensembles de données. Ces données peuvent être des chiffres, des mots, des images, des statistiques **[10].**

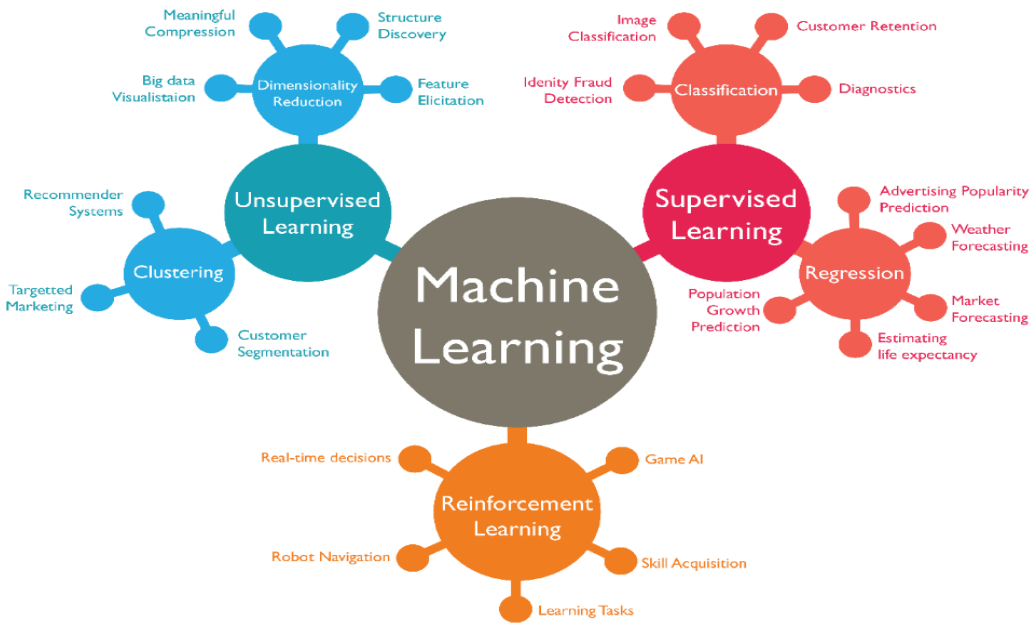

Figure 1.10 : Apprentissage automatique **[11]**

1.3.1.1 Types d'apprentissage automatique

L'apprentissage automatiqueou (***Le machine learning***), fondement de l'intelligence artificielle, englobe un large éventail d'algorithmes adaptés à diverses tâches et ensembles de données. D'une manière générale, ces algorithmes se répartissent en trois principales catégories :

- **Apprentissage supervisé**

L'apprentissage supervisé est défini par l'utilisation de jeux de données étiquetés pour entraîner des algorithmes à classer les données ou à prédire les résultats avec précision. Au fur et à mesure que les données d'entrée sont introduites dans le modèle, celui-ci ajuste ses pondérations jusqu'à ce qu'elles soient ajustées de manière appropriée.

L'apprentissage supervisé permet aux entreprises de résoudre divers problèmes à l'échelle, comme le classement du courrier indésirable dans un dossier autre que leur boîte de réception **[12].**

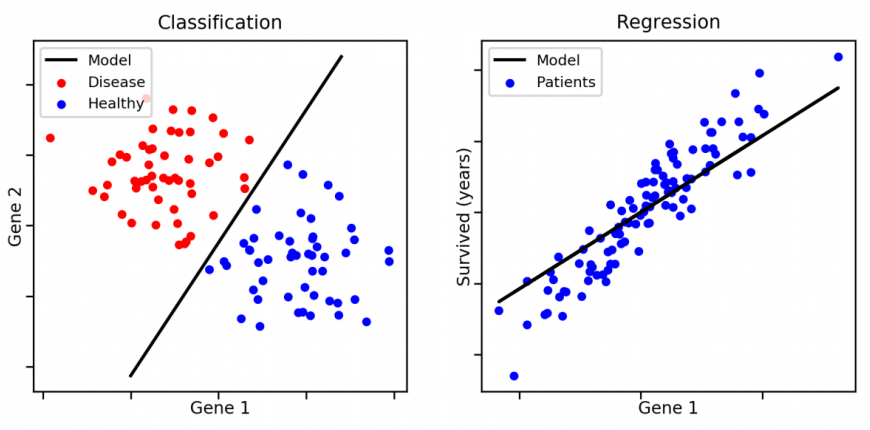

Figure 1.11 : Apprentissage supervisé **[13]**

- **Apprentissage non supervisé**

L'apprentissage non supervisé utilise des algorithmes de ***machine learning*** pour analyser et regrouper des jeux de données non étiquetées (***sous-ensembles appelés clusters***).

Ces algorithmes découvrent des modèles ou des groupes de données cachés sans intervention humaine.

Elle permet également de réduire le nombre de fonctionnalités dans un modèle grâce au processus de réduction de la dimensionnalité **[22].**

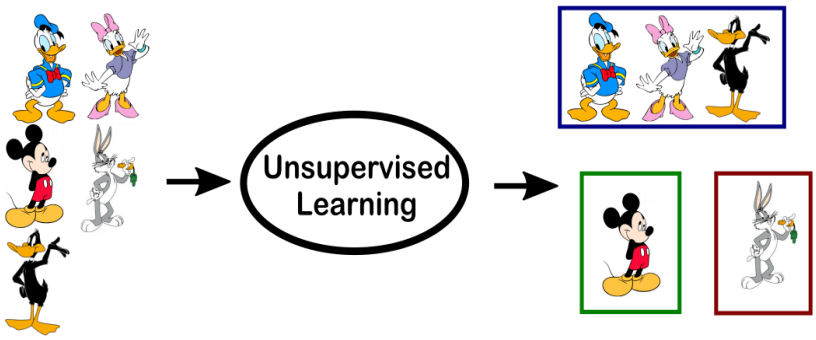

Figure 1.12 : Apprentissage non-supervisé **[13]**

- **Apprentissage semi supervisé**

L'apprentissage semi-supervisé offre un juste milieu entre l'apprentissage supervisé et l'apprentissage non supervisé. Pendant la formation, il utilise un jeu de données étiquetées plus petit pour guider la classification et l'extraction de fonctionnalités à partir d'un jeu de données plus grand et non étiqueté. L'apprentissage semi-supervisé peut résoudre le problème du manque de données étiquetées pour un algorithme d'apprentissage

supervisé. Cela aide également s'il est trop coûteux d'étiqueter suffisamment de données **[22].**

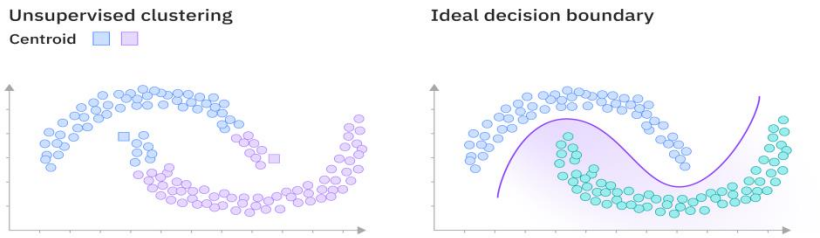

Figure 1.13 : Apprentissage semi supervisé **[24]**

- **Apprentissage par renforcement**

Le ***machine learning*** par renforcement est un modèle de ***machine learning*** similaire à ***l'apprentissage supervisé***, mais l'algorithme n'est pas entraîné à l'aide d'exemples de données. Ce modèle apprend au fur et à mesure en procédant par essais et erreurs. Une séquence de résultats positifs sera renforcée afin d'élaborer la meilleure recommandation ou politique pour un problème donné **[22].**

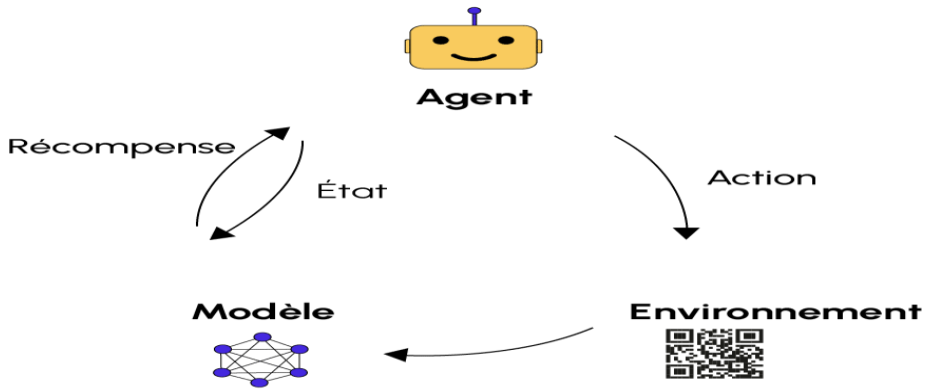

Figure 1.14 : Apprentissage par renforcement**[10]**

1.3.2 Apprentissage profond

Le ***deeplearning*** est une sous-catégorie du ***machine learning*** qui utilise des réseaux neuronaux multicouches, appelés ***réseaux neuronaux profonds***, pour simuler le pouvoir de

décision complexe du cerveau humain. Une forme ou une autre de ***deeplearning*** alimente aujourd'hui la plupart des applications d'intelligence artificielle (IA) que nous utilisons **[22].**

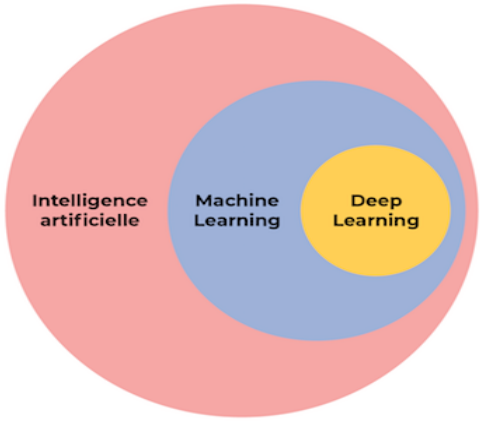

Figure 1.15 : Apprentissage Profond **[14]**

1.3.2.1 Processus d'apprentissage profond

L'apprentissage en profondeur est un sous-ensemble de l'apprentissage automatique qui traite des algorithmes inspirés de la structure et de la fonction des réseaux neuronaux du cerveau humain. Il se caractérise par l'utilisation de réseaux neuronaux profonds,qui sont composés de nombreuses couches de nœuds interconnectés (neurones). Ces réseaux sont entraînés sur de grandes quantités de données pour apprendre des représentations des données par un processus d'extraction de caractéristiques hiérarchiques.

Le processus d'apprentissage en profondeur comprend généralement les étapes suivantes Comme le montre la figure

Figure 1.16 : Processus d'apprentissage profond **[39]**

- **Collecte de données :** Collecte des données pertinentes sur lesquelles le modèle sera entraîné. Il peut s'agir d'images, texte, audio ou tout autre type de données en fonction de la tâche à accomplir **[40].**
- **Prétraitement des données** : Préparer les données pour la formation en les nettoyant, en les normalisant et éventuellement en les augmentant afin d'améliorer les performances du modèle **[40].**
- **Conception du modèle et de l'architecture** : Conception de l'architecture du réseau neuronal, y compris le nombre de couches, les types de couches (par exemple, convolutionnelles, récurrentes) et les connexions entre elles**[40].**
- **Formation :** Utilisation d'un algorithme d'optimisation (tel que la descente de gradient stochastique) pour ajuster les poids du réseau neuronal en fonction des données d'apprentissage **[40].**

- **Validation :** Tester le modèle sur des données inédites pour vérifier sa capacité de généralisation et éviter le***surapprentissage*** **[40].**
- **Test :** Enfin, le modèle formé est testé sur un ensemble de données de test distinct afin d'évaluer ses performances dans des scénarios réels **[40].**

1.3.2.2 Les types d'apprentissage profond

Il existe plusieurs types d'architectures de réseaux neuronaux :

- **Réseaux de neurones convolutifs (CNN)**

Les **CNN** sont des types spécifiques de réseaux neuronaux, composés de couches de nœuds, contenant une couche d'entrée, une ou plusieurs couches masquées et une couche de sortie. Ils sont principalement utilisés dans les applications de vision par ordinateur et de classification des images. Ils peuvent détecter les caractéristiques et les schémas dans les images et les vidéos, permettant l'exécution de tâches telles que la détection d'objets, la reconnaissance d'images, la reconnaissance de formes et la reconnaissance faciale **[22].**

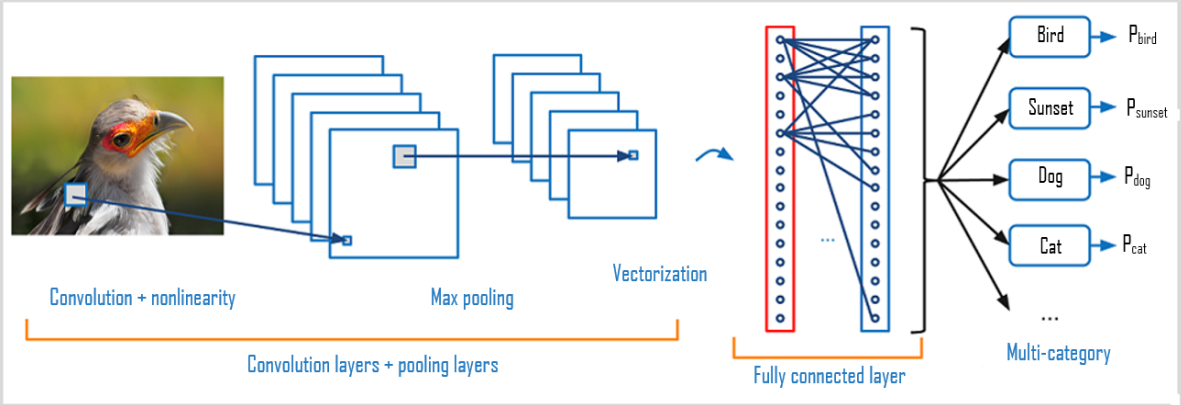

Figure 1.17 : Réseaux de neurones convolutifs (CNN)**[15]**

- **Réseaux antagonistes génératifs (GAN)**

Les ***GANs*** sont des réseaux neuronaux utilisés à la fois dans et en dehors de l'intelligence artificielle (IA) pour créer de nouvelles données ressemblant aux données d'apprentissage d'origine. Il peut s'agir d'images de visages humains, mais d'images générées et non pas du visage de vraies personnes. Le terme « antagoniste » s'explique par les va-et-vient entre les deux parties du GAN : le générateur et le discriminateur **[22].**

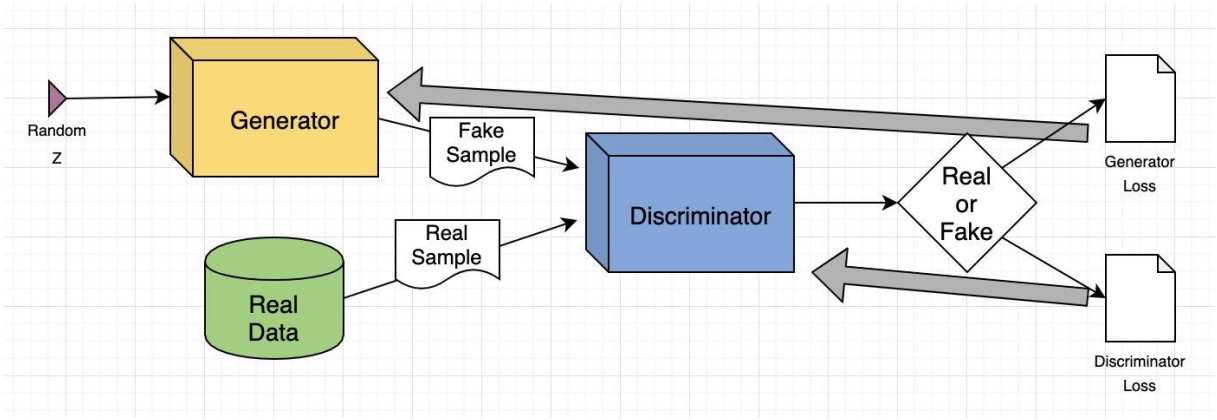

Figure 1.18 : Réseaux antagonistes génératifs (GAN) **[16]**

- **Le générateur :** prend un **vecteur de bruit aléatoire** comme entrée et génère une donnée. Il a pour but de produire des données fausses ressemblant le plus possible aux réelles **[33]**.
- **Le discriminateur** : reçoit soit une donnée réelle, soit une donnée générée par le générateur et doit faire la **distinction entre les deux**. Il sort alors une probabilité indiquant si la donnée est réelle ou générée **[17].**

➢ **Réseaux de neurones récurrents (RNN)**

Les ***RNN***s utilisent leur « mémoire » : les informations des entrées antérieures ont une influence sur l'entrée et la sortie en cours.Sont généralement utilisés dans les applications de traitement du langage naturel et de reconnaissance automatique de la parole, car ils utilisent des données séquentielles ou de séries temporelles **[22].**

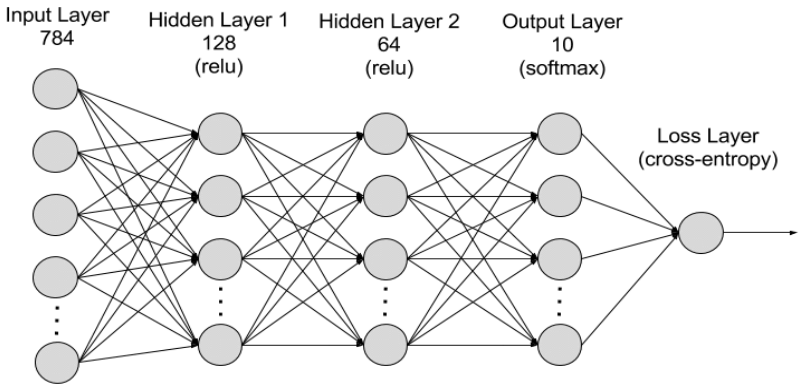

Figure 1.19 : Réseaux de neurones récurrents **[16]**

- **Conditional Generative Adversarial Network (CGAN)**

Les ***GANs*** conditionnels sont une extension des ***GAN*** traditionnels qui introduisent une couche supplémentaire de conditionnement à la fois au générateur et au discriminateur. Dans une configuration ***GAN***typique, un générateur crée des images à partir de bruit aléatoire et un discriminateur fait la distinction entre les images réelles et générées. Dans les ***CGAN***, les deux réseaux sont conditionnés par des informations auxiliaires, telles que des étiquettes de classe ou des descriptions textuelles, guidant le processus de génération vers la production de types spécifiques d'images **[18].**

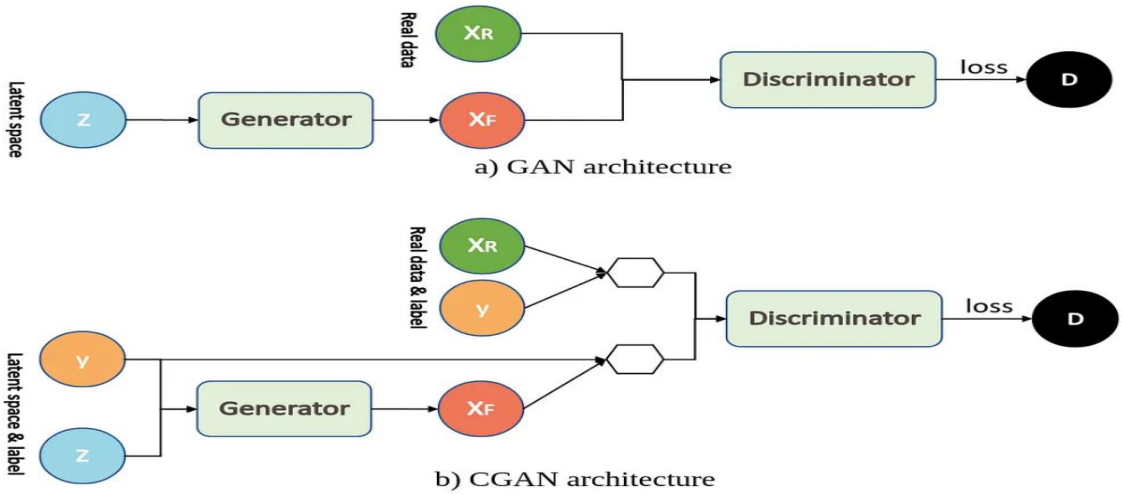

Figure 1.20: Conditional Generative Adversarial Network **[19]**

1.3.3 Transformateurs de vision

Les transformateurs de vision (***VIT***) sont des architectures avancées d'apprentissage profond qui transforment les tâches de vision par ordinateur en offrant des performances impressionnantes, en capturant des informations globales et en gérant efficacement les dépendances à long terme, ce qui entraîne des avancées significatives dans le domaine de l'analyse d'images.

L'idée principale est de traiter les données d'image comme une séquence de zones, ou de régions, et d'utiliser des mécanismes d'attention pour capturer les relations entre les régions afin de faire une prédiction. Concentrons-nous sur les deux principales technologies à la base des ***VIT*** [20].

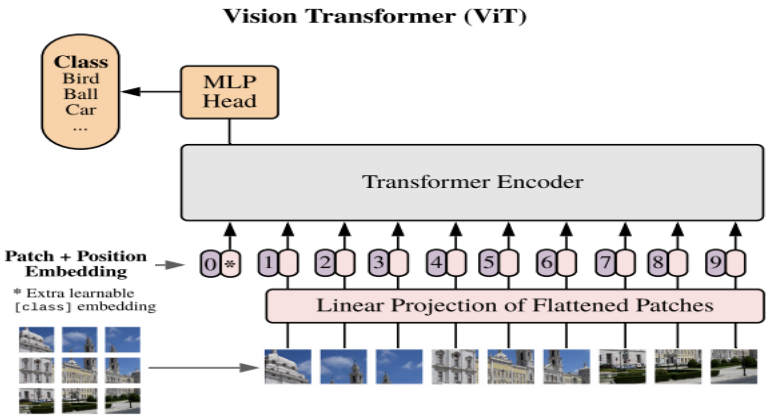

Figure 1.21 : Transformateurs de vision **[21]**

1.3.3.1 Les types de transformateurs par vision

Ce sont des modèles avec leurs études qu'explorent l'efficacité des modèles Vision Transformer ***(ViT)*** dans le domaine de la vision par ordinateur. Les résultats présentés dans le tableau ci-dessous démontrent la capacité des ***ViT***, en particulier ceux pré-entraînés sur de vastes ensembles de données, à être transférés avec succès à travers diverses tâches de vision. Cette adaptabilité souligne leur polyvalence et leur potentiel à surpasser les approches convolutionnelles traditionnelles, établissant ainsi une nouvelle référence dans le traitement de la vision.

Modèle	Paramètres (Millions)	Taille de l'Image	Précision Top-1 (ImageNet)	Ensembles de Données Utilisés pour le Pré-entraînement	Caractéristiques Spéciales
ViT-Base	86	384×384	77.9%	ImageNet	Modèle ViT original
ViT-Large	307	384×384	76.5%	ImageNet	Version plus grande de ViT-Base
ViT-Huge	632	384×384	77.0%	ImageNet	Le plus grand modèle ViT
DeiT-Small	22	224×224	79.8%	ImageNet	Token de distillation pour l'efficacité de l'entraînement
DeiT-Base	86	224×224	81.8%	ImageNet	Version plus grande de DeiT-Small
Swin Transformer	88	224×224	83.0%	ImageNet-21k, ImageNet	Architecture hiérarchique, efficacité améliorée

Tableau 1.1 : Les types de transformateurs par vision.

1.4 Conclusion

Dans ce chapitre, nous donnons un aperçu des maladies du blé et de leurs méthodes de diagnostic traditionnelles et modernes, ainsi que passons en revue les concepts de base dans le domaine de l'intelligence artificielle et de l'apprentissage profond, qui sont la pierre angulaire de notre application de détection des maladies du blé. Dans le chapitre suivant, nous présenterons une synthèse approfondie des modèles d'intelligence artificielle et des applications mobiles existantes dédiées au diagnostic des maladies foliaires du blé, afin de mieux situer notre propre approche dans l'état de l'art.

Chapitre2

Synthèse des modèles d'IA et des applications mobiles pour le diagnostic des maladies du blé

2.1. Introduction

Le blé figure parmi les cultures céréalières les plus cruciales au niveau global, tant pour la sûreté alimentaire que pour l'économie du secteur agricole. Cependant, sa culture est susceptible à différentes maladies foliaires comme la rouille, l'oïdium ou la ***septoriose***, qui peuvent causer des réductions importantes de productivité. Confrontée à ces menaces, l'intelligence artificielle (***IA***) apparaît comme une option prometteuse pour automatiser et perfectionner le diagnostic des maladies basé sur les images de feuilles.

Ce chapitre présente un récapitulatif des recherches précédentes dans ce secteur, en soulignant les modèles d'apprentissage profond les plus couramment employés, leurs performances ainsi que leurs contraintes. Nous traitons aussi les stratégies clés de prétraitement et d'augmentation de données, essentielles pour rehausser la qualité des images et pallier au déficit de jeux de données équilibrés.

De plus, une analyse comparative des applications mobiles actuelles est effectuée pour repérer les caractéristiques communes, les lacunes du marché et les possibilités d'amélioration. Pour conclure, nous présentons notre propre apport en développant un modèle qui allie la segmentation sémantique, la création d'images à l'aide des ***GANs***, la classification par Vision Transformer ***(ViT),*** et son déploiement sur une plateforme mobile destinée aux professionnels du secteur agricole.

2.2 Travaux Scientifiques Connexes

Dans cette section, nous analyserons les recherches antérieures liées au diagnostic automatique des maladies des feuilles de blé à l'aide de techniques d'IA.Le tableau 2.1ci-dessousprésente une comparaison de quatre modèles de ***Deep Learning*** largement utilisés dans le diagnostic des maladies des feuilles de blé. Voici les points clés de chaque modèle :

- **DenseNet121** : Ce modèle utilise des connexions denses, favorisant une réutilisation efficace des caractéristiques. Il présente des performances solides, en particulier lorsqu'il est associé à ***CycleGAN***. Toutefois, sa consommation de mémoire est plus élevée et il pourrait avoir des difficultés à généraliser à des images autres que celles des feuilles de blé.
- **ResNet50V2** : Avec ses connexions de raccourci, ce modèle facilite l'entraînement des réseaux profonds. Il offre de bonnes performances, particulièrement en combinaison avec ***ADASYN***. Cependant, il nécessite un volume de données plus important pour un entraînement optimal par rapport aux modèles plus simples.
- **Xception** : Ce modèle utilise des convolutions séparables en profondeur, ce qui réduit le nombre de paramètres tout en améliorant l'efficacité du calcul. Bien qu'il offre de bonnes performances avec ***CycleGAN***, il pourrait être moins performant que des modèles plus complexes lorsqu'il s'agit de traiter des motifs très complexes.
- **MobileNetV2** : Conçu spécifiquement pour les appareils mobiles, il est particulièrement adapté aux environnements aux ressources limitées. Grâce à son architecture optimisée, il affiche de bonnes performances, notamment avec ***CycleGAN*** et ***ADASYN***, mais peut-être moins précis par rapport à des modèles plus grands et plus complexes.

Modèle	Caractéristiques architecturales principales	Points forts	Performance dans l'étude actuelle	Faiblesses potentielles
DenseNet121[22]	Connexions denses entre les couches. Chaque couche est directement connectée à toutes les couches suivantes.	Réutilisation efficace des caractéristiques, atténuation du problème de disparition du gradient.	Bonnes performances, surtout avec CycleGAN.	Peut-être plus gourmand en mémoire en raison des connexions denses. Peut avoir du mal à généraliser à des types d'images différents des feuilles de blé.
ResNet50V2[23]	Connexions de raccourci (skip connexion) permettant aux donnés de contourner certaines couches.	Facilite l'entrainement des réseaux profonds, les unités résiduelles apprennent des fonctions résiduelles.	Bonnes performances, surtout avec ADASYN.	Peut nécessiter plus de donnés pour un entrainement efficace par rapport aux modèles plus petits.
Xception[24]	Couches de convolution séparables en profondeur, architecture inspirée d'Inception.	Réduction de nombre du paramètres, efficacité de calcul élevée, capture de caractéristiques à différentes échelles.	Bonnes performances, surtout avec CycleGANmalgré quelques erreurs mineurs.	Bien qu'efficace, peut ne pas etre aussi performant que des modèles plus grands pour capturer des motifs très complexes.
MobileNetV2[25]	Conçu spécifiquement pour les appareils mobiles et les environnements aux ressources limitées.	Convolution séparable en profondeur, architecture Bottleneck inversée, raccourcis linéaires.	Bonnes performances, surtout avec CycleGAN et ADASYN.	Etant donné qu'il est conçu pour des ressources limitées, il peut etre moins précis dans certains cas par rapport à des modèles plus grands et plus complexes.

Tableau 2.1 : Modèles d'IA pour le diagnostic des maladies des feuilles de blé

Suite à l'examen des modèles de ***Deep Learning*** dans cette section, la prochaine partie explore comment l'utilisation de divers ensembles de données et des techniques d'augmentation, telles que ***CycleGAN*** et ***ADASYN***, peut améliorer l'efficacité de ces modèles pour le diagnostic des maladies des feuilles de blé.

2.2.1 Exploitation des ensembles de données et des techniques d'augmentation pour le diagnostic des maladies foliaires du blé

2.2.1.1 Utilisation de divers ensembles de données

Les chercheurs exploitent plusieurs ensembles de données pour l'entraînement des modèles de classification des maladies du blé. Parmi eux :

- Données originales utilisées sans augmentation, bien que limitées en taille.
- Données augmentées avec des techniques avancées comme ***CycleGAN*** et ***ADASYN***.
- Comparaison des résultats avec différentes techniques d'augmentation pour identifier les plus efficaces.
  Cela reflète une tendance à surmonter les limitations des petits ensembles de données en agriculture par des méthodes d'augmentation générative.

2.2.1.2 Progrès dans les techniques d'intelligence artificielle

Les modèles testés incluent différentes architectures de réseaux de neurones profonds :

- **CNN** classiques : ***ResNet50V2, DenseNet121, MobileNetV2, Xception***.
- Techniques avancées : utilisation de ***CycleGAN***et***ADASYN*** pour améliorer la performance des modèles face à des données limitées et déséquilibrées.
- Impact des augmentations : l'étude montre que ***MobileNetV2*** atteint une précision sur les ensembles de données augmentés avec ***CycleGAN***et***ADASYN***, soulignant l'importance des stratégies de prétraitement et de génération de données.

2.2.1.3 Techniques de prétraitement

Les étapes de prétraitement sont cruciales pour améliorer la qualité des images avant l'entraînement :

- Normalisation et redimensionnement pour homogénéiser les entrées du modèle.

- Techniques spécifiques de nettoyage d'image pour éliminer le bruit et améliorer la clarté des caractéristiques pathologiques des feuilles de blé.
- Segmentation dans certains cas, bien que cette technique ne soit pas toujours nécessaire pour les modèles ***CNN***.

2.2.1.4 Segmentation

Certains travaux intègrent des méthodes de segmentation pour mieux délimiter les zones affectées des feuilles :

- ***CycleGAN***et ***ADASYN*** permettent d'augmenter les images et d'améliorer la différenciation des classes.
- Méthodes traditionnelles de classification et d'extraction de caractéristiques, bien que moins efficaces que les modèles basés sur l'apprentissage profond.

2.2.1.5 Augmentation des données

L'augmentation des données est divisée en deux catégories :

- Techniques classiques : rotations aléatoires, zooms, translations et flips, utilisées pour augmenter la diversité des images d'entraînement et améliorer la robustesse des modèles.
- Techniques avancées :
    - ***CycleGAN*** : génère des images réalistes et variées pour enrichir les classes sous-représentées.
    - ***ADASYN***: technique d'***oversampling*** adaptatif pour rééquilibrer les classes et améliorer la précision sur les cas rares.

2.3 Applications Mobiles Similaires Existantes

Maintenant, nous examinons le tableau comparatif, qui donne un aperçu des différentes applications similaires à notre concept. Notre recherche sur l'App Store et Google Play a mené à la sélection d'applications pertinentes pour les maladies des feuilles de blé, offrant un téléchargement gratuit, complètes en programmation, multilingues (avec priorité à l'arabe), et diverses dans leurs capacités de diagnostic.

Un code couleur **(✓ et ✕)** indique si chaque fonctionnalité est présente dans l'application. Nous soulignerons les lacunes pour justifier les améliorations de notre application.

Nom de l'application	**Plantix**	**Agrio**	**Agri IA**	**Dr Nabat**
Types de maladies des feuilles traitées	Rouille du blé, Oïdium, Fusariose, Mildiou, Tache foliaire, Necrose, Carences nutritionnelles	Mildiou du blé, Rouille brune, Septoriose, Fusariose, Oïdium, Charbon nu	Rouille de blé, Mildiou, Fusariose, Oïdium, Tache foliaire, Carences nutritionnelles	Maladies végétales dont la Rouille, la Septoriose, la Fusariose, le Mildiou et l'Oïdium 21
Interface conviviale	✓	✓	✓	✓
Informations médicales	✓	✓	✓	✓
Système d'exploitation	iOS,Android	iOS,Android	iOS,Android	Android, Web
Profils d'utilisateurs & Historique	✕	✕	✕	✕
Support multilingue	✓ (Plusieurs langues)	✓	✓ (Arabe, Français, Anglais)	✓ (Arabe, Anglais)
Intégration aves des agronomes	✓	✓	✓	✕
Forums communautaires	✓	✓	✓	✕
Sécurité des données	✕	✕	✓	✕
Notifications et Rappels	✓	✓	✓	✕
Intégration avec des appareils portables	✕	✕	✕	✕
Prix de l'abonnement	Gratuit	Gratuit avec options payantes	Gratuit avec options payantes	Gratuit

Tableau 2.2 : Tableau comparatif détaillés des applications mobiles de diagnostic des maladies de blé

Dans chaque élément de la colonne du tableau illustré dans le **tableau 2.2**, nous observons que:

- **Types de maladies des feuilles traitées:*Plantix*** couvre les maladies des feuilles de diverses cultures, y compris le blé. ***Agrio*** utilise l'IA pour la détection des maladies des plantes, y compris celles du blé. ***ICBA*** App identifie les maladies et carences nutritionnelles des cultures, dont le blé. Dr. ***Nabat***détecte 21 maladies végétales, incluant celles affectant le blé.
- **Interface Conviviale :** Toutes les applications (***Plantix, Agrio, ICBA App, Dr. Nabat***) sont signalées comme ayant une interface conviviale performante (✓).
- **Informations Médicales :** Toutes les applications fournissent des informations (✓), mais leur exactitude n'est pas spécifiée.
- **Système d'exploitation :*Plantix, Agrio et ICBA*** App sont disponibles sur ***iOS***et ***Android.Dr. Nabat*** est aussi accessible via le Web.
- **Profils d'utilisateur et Historique :** Aucune des applications (***Plantix, Agrio, ICBA App, Dr. Nabat***) ne permet la création de comptes utilisateurs ni la consultation des diagnostics précédents (✕).
- **Support Multilingue :*Plantix*** supporte plusieurs langues (✓). ***ICBA App et Dr. Nabat*** supportent l'arabe, le français et l'anglais (✓).
- **Intégration avec des agronomes :*Plantix et Agrio*** offrent un contact avec des agronomes (✓), tandis ***qu'ICBA App et Dr. Nabat*** n'offrent pas cette fonctionnalité (✕).
- **Forums communautaires ou Groupes de discussion :** Aucune des applications (***Plantix, Agrio, ICBA App, Dr. Nabat***) n'offre de forums ou de groupes de discussion (✕).
- **Confidentialité et Sécurité :** Aucune des applications ne protège suffisamment la confidentialité et la sécurité des utilisateurs (✕).
- **Notifications et rappels :*Plantix*** et ***Agrio***incluent des rappels et des notifications (✓), tandis que ***ICBA*** App et ***Dr. Nabat*** ne le font pas (✕).

- **Intégration avec les appareils portables :** Aucune des applications n'intègre d'appareils portables (✕).
- **Prix de l'abonnement :*Plantix*** est gratuit, ***Agrio*** est gratuit avec options payantes, et le modèle de tarification des autres n'est pas spécifié.

2.4 Contribution Proposée au Diagnostic des Maladies des Feuilles de Blé

Après avoir analysé les travaux précédents qui ont contribué au diagnostic des maladies du blé à partir d'images de feuilles, nous clarifions notre contribution et la résumons comme suit :

- **Améliorer la qualité des données** en prétraitant les images par normalisation, redimensionnement, réduction du bruit et amélioration du contraste.
- **Développer une architecture Vision Transformer (*ViT*)** pour la classification des maladies des feuilles de blé.
- **Évaluer l'efficacité de notre approche** en comparant nos résultats avec les études existantes sur la classification des maladies du blé.
- **Réaliser une application mobile** qui facilitera l'utilisation de notre modèle d'IA et le rendra accessible aux agriculteurs et aux spécialistes agricoles.

2.5 Conclusion

Ce chapitre a proposé une revue approfondie des différentes approches d'intelligence artificielle utilisées pour le diagnostic des maladies foliaires du blé. Il a notamment mis en évidence le rôle des modèles d'apprentissage profond tels que les réseaux de neurones convolutifs***(CNN)*** et les ***Transformers***, ainsi que l'importance des techniques d'augmentation de données pour améliorer la robustesse des modèles.Une analyse comparative des principales applications mobiles existantes dédiées à cette tâche a permis de souligner leurs principales limites, notamment en termes de précision, de diversité des maladies détectées, d'ergonomie utilisateur et de dépendance à la connectivité. Ces constats mettent en évidence la nécessité d'un système plus performant, plus accessible et mieux adapté aux besoins réels des agriculteurs.

Le prochain chapitre sera donc consacré à la description détaillée de notre propre solution, qui vise à combler ces lacunes, un pipeline de prétraitement optimisé et une interface mobile robuste.

Chapitre 3

Conception du système

3.1 Introduction

Le blé est l'une des cultures stratégiques les plus importantes à l'échelle mondiale. Les maladies qui l'affectent ont un impact significatif sur la productivité et la qualité, entraînant ainsi d'importantes pertes économiques. Les avancées majeures dans le domaine de l'intelligence artificielle, en particulier l'apprentissage profond, ont permis le développement d'outils innovants pour soutenir l'agriculture de précision, notamment dans la détection précoce des maladies des plantes. Ce chapitre présente une vue d'ensemble de la conception du système proposé pour identifier et classifier les maladies courantes du blé à partir d'images de feuilles ou d'épis infectés.

D'abord, nous commencerons par discuter de l'architecture générale du modèle de classification proposé. Ensuite, nous examinerons en détail les fonctions spécifiques du système, notamment la collecte et la préparation des données, le prétraitement, l'entraînement du modèle, l'évaluation de ses performances, jusqu'à son déploiement pour une utilisation pratique.

3.2 Architecture générale du système

L'architecture générale décrit le processus de classification des images des maladies du blé. Le processus commence par la collecte d'un ensemble de données (***Dataset***) comprenant des images de plants de blé (***feuilles, épis***) présentant divers symptômes de maladies, ainsi que des images de plants sains. Ces images sont ensuite soumises à une phase de prétraitement (***Pre-processing***) visant à améliorer leur qualité et à les uniformiser.

Les données prétraitées sont ensuite divisées en trois ensembles : entraînement (***Training***), validation (***Validation***) et test (***Testing***). L'ensemble d'entraînement est utilisé pour apprendre au modèle à reconnaître les motifs caractéristiques de chaque maladie et à extraire les caractéristiques importantes.

Une fois le modèle entraîné, ses performances sont évaluées à l'aide des ensembles de validation et de test afin de garantir sa précision et sa capacité à généraliser à de nouvelles données. Sur la base de cette analyse, une décision est prise : si le modèle répond aux critères de performance requis, il est validé et déployé (par exemple, dans une application web ou mobile). Sinon, il est soumis à des phases supplémentaires d'entraînement et d'optimisation.

Ce processus itératif permet de développer un système robuste et efficace pour l'identification et la classification des maladies du blé.Comme illustré à la ***figure 3.1***.

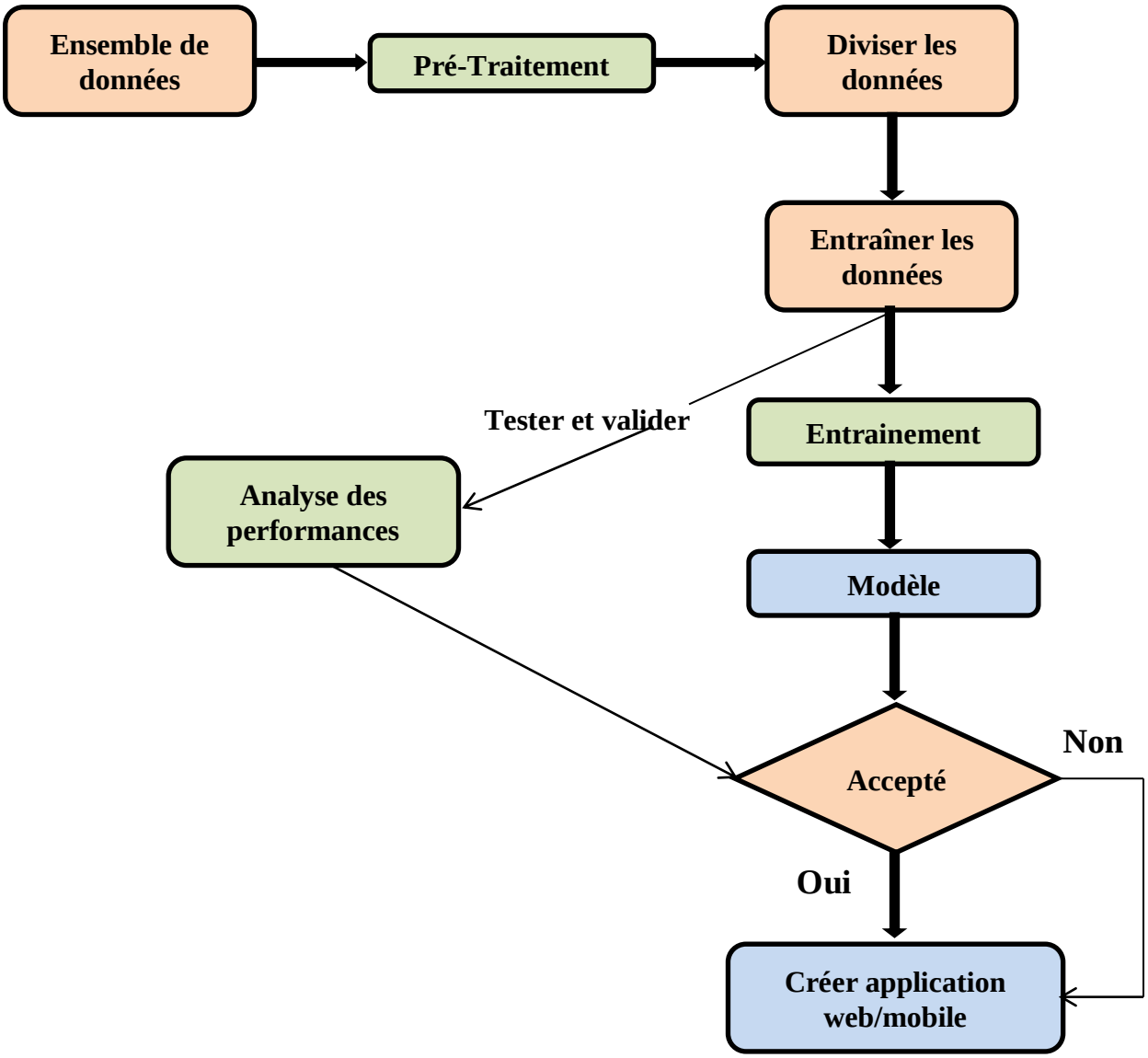

Figure 3.1 : Architecture générale du système de classification des maladies du blé

3.3 L'architecture détaillée du système

Cette section explique l'architecture globale et les procédures suivies pour la gestion de l'ensemble de données, depuis la collecte jusqu'au prétraitement, en passant par les détails liés à l'entraînement, l'évaluation et le déploiement du modèle. L'objectif est de garantir que les données soient suffisamment préparées pour entraîner les modèles d'apprentissage automatique et atteindre les meilleures performances possibles.

3.3.1 Collecte des données (Data Gathering)

L'ensemble de données collecté est constitué d'images numériques (***au format JPG ou PNG***) représentant des feuilles et des épis de blé. Ces images ont été obtenues à partir de

sources variées, telles que des champs agricoles, des serres, ainsi que des bases de données publiques comme ***PlantVillage.***

L'ensemble comprend des dossiers distincts pour chaque catégorie de maladies connues (***par exemple*** : rouille de la tige, rouille des feuilles, oïdium, septoriose), ainsi qu'un dossier dédié aux plantes saines (***Healthy*****)**.

La diversité des conditions de prise de vue a été prise en compte — éclairages variés, arrière-plans différents, stades d'infection multiples — afin d'accroître le réalisme et la robustesse des données.

3.3.2 Prétraitement des données (*Pre-processing*)

La phase de prétraitement est une étape critique dans les tâches de classification d'images, où nous modifions les données par des processus tels que le redimensionnement, l'application de transformations et la normalisation avant de les transmettre au modèle d'entraînement. Dans notre système, nous avons redimensionné toutes les images à **224x224** pixels et effectué une normalisation ***(voir Figure 3.2).***

- **Redimensionnement (transforms.Resize((224, 224))):** Les images de l'ensemble de données original peuvent avoir des dimensions variables. Pour standardiser ces dimensions, nous les redimensionnons en (224x224 pixels). Le modèle ***ViT*** utilisé attend cette taille d'entrée.

- **Normalisation (transforms.Normalize([0.5]*3, [0.5]*3)):** Après le redimensionnement (et d'autres augmentations pour l'ensemble d'entraînement), les images sont normalisées. Ce processus consiste à mettre à l'échelle les valeurs des pixels (généralement converties en tenseurs dans la plage [0, 1] par **transforms.ToTensor())** pour qu'elles aient une moyenne de 0.5 et un écart-type de 0.5 pour chaque canal de couleur (Rouge, Vert, Bleu). La normalisation aide à accélérer la convergence du processus d'entraînement et améliore les performances globales du modèle.

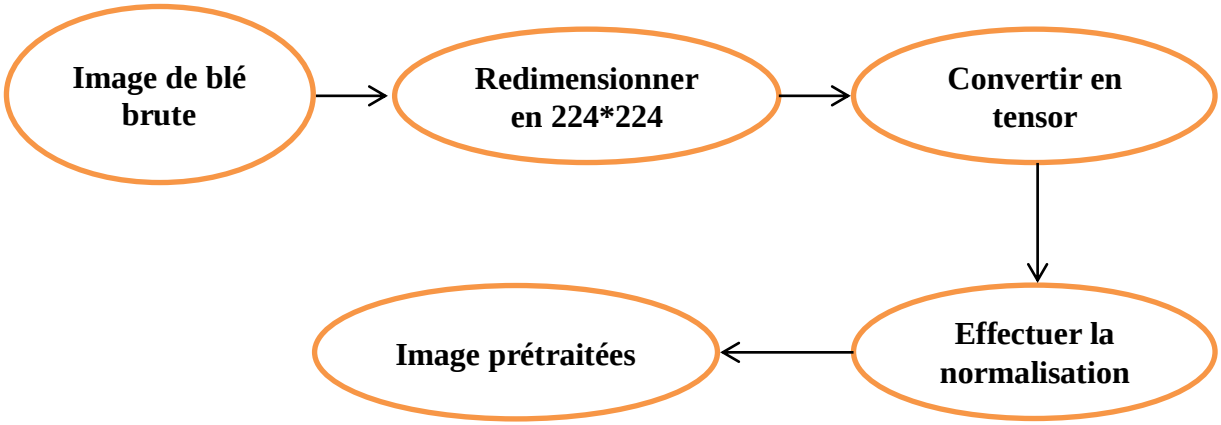

Figure 3.2 : Les étapes de Prétraitement des données

3.3.3 Division de l'ensemble de données (*Splitting the Dataset*)

Dans notre approche, nous chargeons directement des ensembles de données distincts pour l'entraînement, la validation et le test à partir des répertoires spécifiés (*train_dir*, *valid_dir*, *test_dir*).
Le code utilise ***ImageFolder*** pour charger les images de ces répertoires. Ensuite, des DataLoader sont créés pour chaque phase ***(train, valid, test).*** Ces ***DataLoader*** gèrent le chargement des données par lots ***(batchs***), le mélange des données pour l'ensemble d'entraînement (***shuffle=True***), et l'utilisation de workers parallèles pour accélérer le chargement (***num_workers=2***) ***(Voir figure 3.3)*** **.**

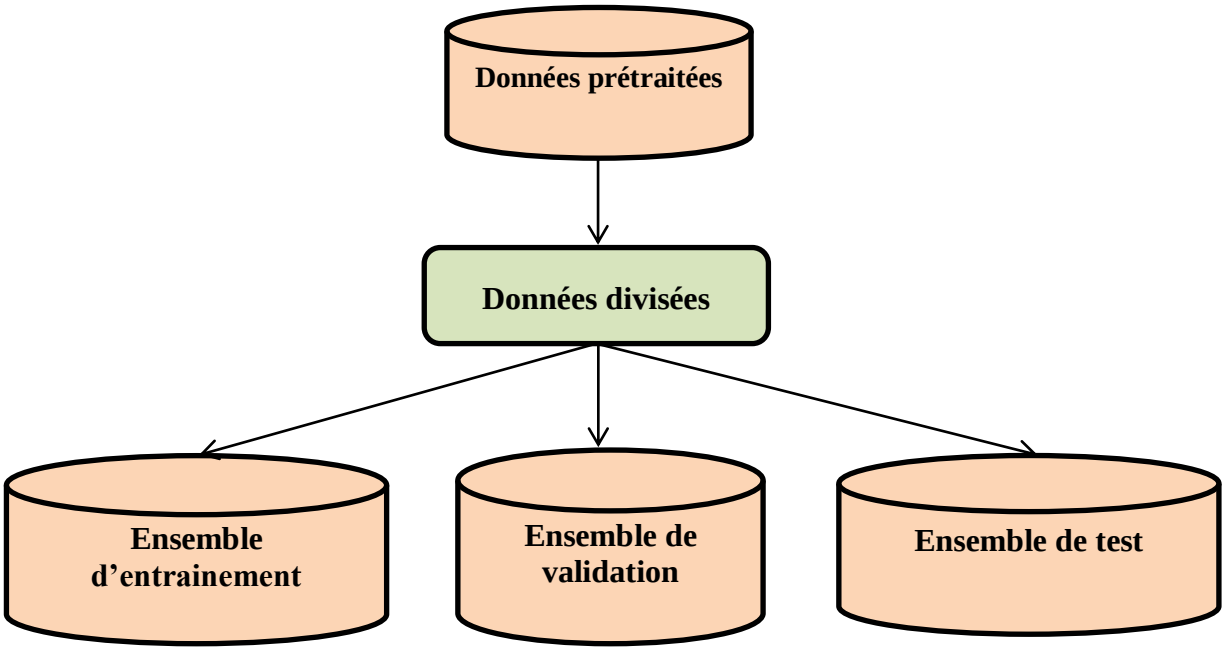

Figure 3.3 : Partitionnement des données.

3.3.4 Entraînement du modèle (*Model Training*)

Dans l'apprentissage profond, la phase d'entraînement est essentielle pour développer un modèle capable de faire des prédictions précises (comme le diagnostic des maladies du blé) sur de nouvelles données. Cette phase comprend plusieurs étapes, commençant par la préparation des données d'entraînement et se terminant par l'obtention du modèle entraîné (***Model Fitting***). L'augmentation des données (***Data Augmentation***) est une technique clé dans ce processus, utilisée pour augmenter artificiellement la taille du jeu de données d'entraînement en créant des copies modifiées des images de feuilles et d'épis de blé existants. Cela améliore les performances du modèle en le rendant plus robuste face aux variations des données (comme les différentes conditions d'éclairage, les multiples angles de prise de vue, ou les stades précoces d'infection). La structure du modèle pour la phase d'entraînement inclut la collecte des données d'entraînement, l'application des techniques d'augmentation des données, l'utilisation des données augmentées (***Augmented Data***) pour l'entraînement, et enfin l'obtention du modèle final entraîné (***Fitted Model***).

Figure 3.4 : Architecture de la phase d'entraînement

3.3.4.1 Augmentation des Données (*Data Augmentation*)

L'augmentation des données est une technique utilisée pour augmenter la taille et la diversité des données d'entraînement pour les réseaux d'apprentissage profond, ce qui aide à améliorer les performances des réseaux neuronaux en réduisant le surajustement et en améliorant la généralisation. Notre code applique les transformations suivantes spécifiquement à l'ensemble d'entraînement ***(transform_train)*** :

- **Retournement Horizontal Aléatoire (transforms.RandomHorizontalFlip()):** Retourne l'image horizontalement avec une probabilité (généralement 0.5 par défaut).
- **Rotation Aléatoire (transforms.RandomRotation(15)):** Fait pivoter l'image d'un angle aléatoire dans la plage de +/- 15 degrés.
- **Retournement Vertical Aléatoire (transforms.RandomVerticalFlip()):** Retourne l'image verticalement.
- **Variation de Couleur (transforms.ColorJitter(brightness=0.3, contrast=0.3, saturation=0.3)):** Modifie aléatoirement la luminosité, le contraste et la saturation de l'image.

- **Transformation Affine Aléatoire (transforms.RandomAffine(degrees=0, translate=(0.05, 0.05))):** Applique une translation aléatoire jusqu'à 5% de la largeur/hauteur de l'image.
- **Perspective Aléatoire (transforms.RandomPerspective(distortion_scale=0.2, p=0.5)):** Applique une transformation de perspective aléatoire avec une probabilité de 0.5.

Ces techniques aident le modèle à apprendre des caractéristiques invariantes et à mieux généraliser à des données non vues.

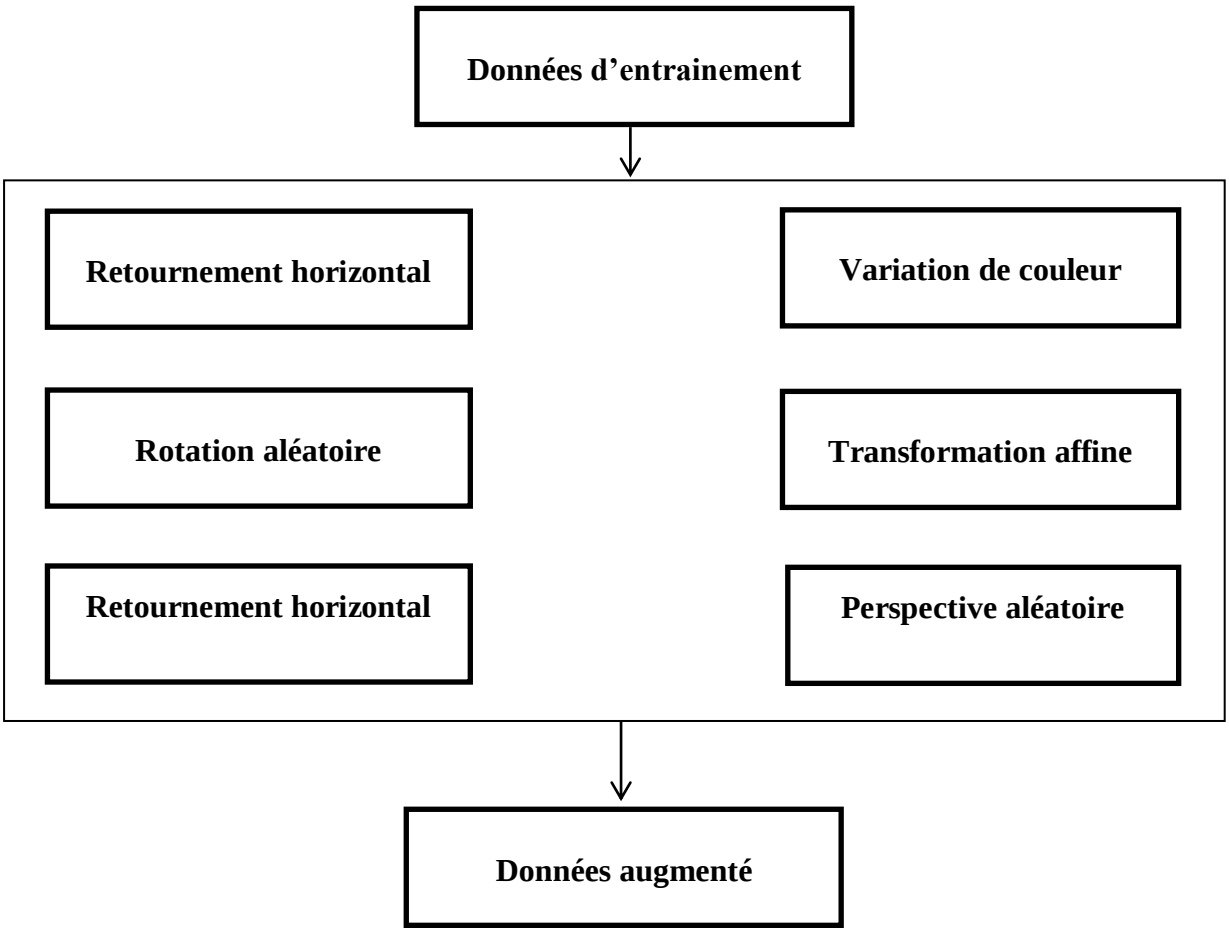

Figure 3.5 : Processus d'augmentation des données

3.3.4.2 Modèles Utilisés (*UsedModels*)

Dans ce travail, nous utilisons ***VIT*.** Ce modèle est présenté dans les sous-sections suivantes:

Modèles Vision Transformer (*ViT*)

Les modèles Vision Transformer ***(ViT)*** représentent une technologie récente et prometteuse que nous souhaitons explorer dans le cadre de cette recherche sur la détection des maladies du blé. Cette technique fait partie des évolutions continues dans le domaine de l'intelligence artificielle et de l'apprentissage profond. Elle est principalement utilisée pour analyser de grands ensembles de données et améliorer la précision des prédictions ainsi que les capacités de traitement.

Dans cette étude, nous nous concentrons sur l'utilisation d'un modèle Vision Transformer ***(ViT)*** pré-entraîné. Spécifiquement, nous utilisons ***ViTForImageClassification*** de la bibliothèque ***transformers*** de ***Hugging*** Face, chargé avec les poids pré-entraînés de ***'google/vit-base-patch16-224-in21k'***. La tête de classification du modèle (la dernière couche linéaire) est remplacée pour correspondre au nombre de classes de notre ensemble de données ***(num_classes = 15).*** Le modèle est ensuite transféré sur le périphérique de calcul spécifié ***(device).***

Le code utilise également une fonction de perte ***CrossEntropyLoss*** pondérée ***(weight=class_weights).*** Les poids sont calculés pour contrer le déséquilibre des classes dans l'ensemble de données, donnant plus d'importance aux classes sous-représentées pendant l'entraînement. L'optimiseur choisi est ***AdamW*** et un ordonnanceur de taux d'apprentissage ***CosineAnnealingLR*** est utilisé pour ajuster le taux d'apprentissage pendant l'entraînement.
La boucle d'entraînement itère sur les epochs, effectue une passe avant, calcule la perte, effectue une rétropropagation (uniquement pour la phase d'entraînement) et met à jour les poids du modèle. Le meilleur modèle est sauvegardé en fonction de la perte de validation la plus faible.

L'architecture générale d'un Vision Transformer est illustrée ci-dessous

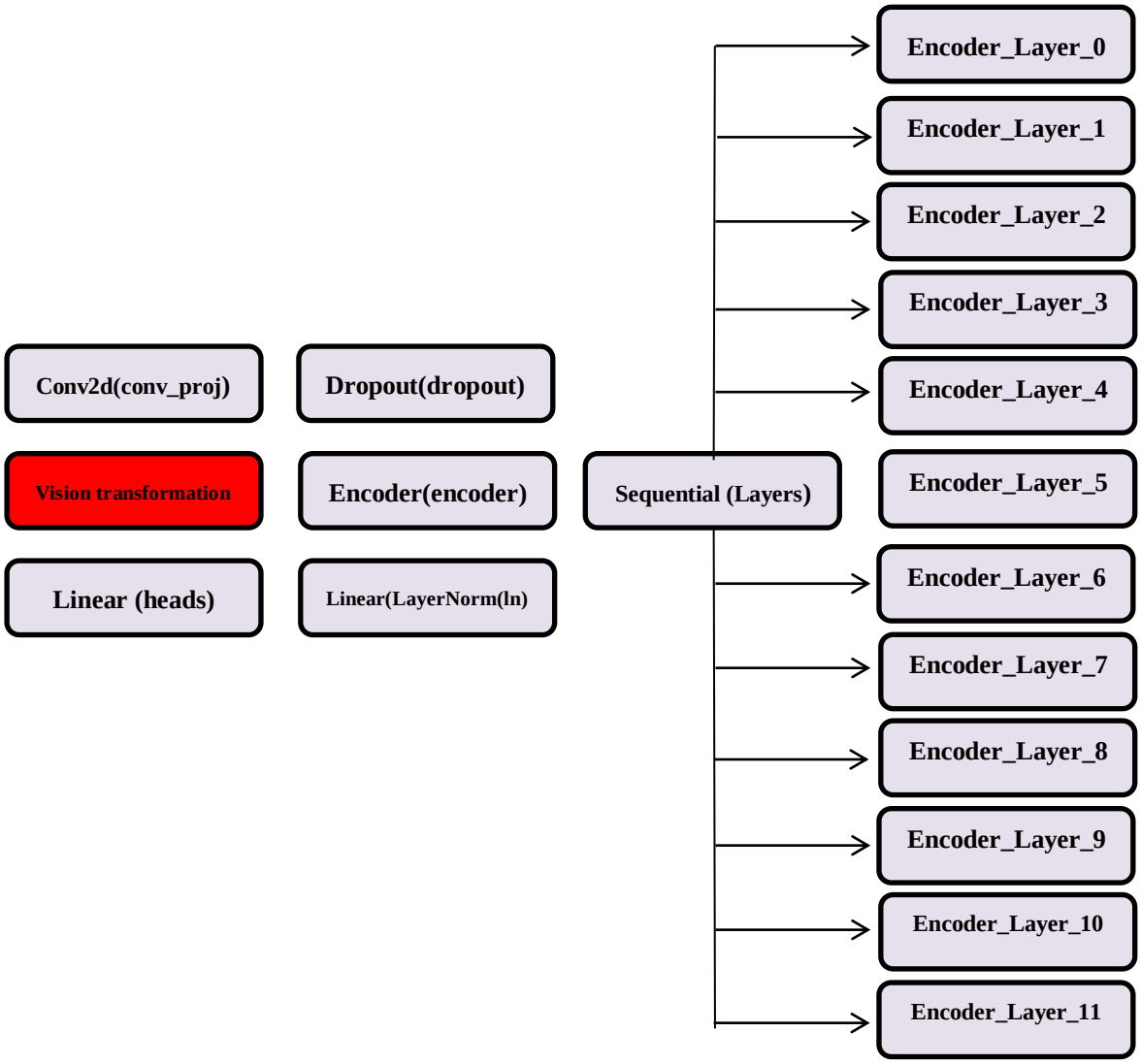

Figure 3.6 : Architecture du Vision Transformer

3.3.5 Test du Modèle (*Model Testing*)

La phase de test est l'étape au cours de laquelle le modèle entraîné est évalué de manière finale à l'aide d'un **Ensemble de test (*Test Set*)** – c'est-à-dire des données que le modèle n'a jamais vues pendant l'entraînement ou la validation – afin de mesurer sa performance réelle.
Cette étape est cruciale, car elle constitue ***l'évaluation finale*** avant le déploiement du modèle dans un ***environnement réel*** (comme un champ ou une application destinée aux agriculteurs).

Pour obtenir une ***prédiction de la classe de la maladie*** (ou pour déterminer si la plante est saine), on ***fournit une image*** de la plante de blé (***feuille ou épi***), ***ayant les mêmes dimensions*** que celles utilisées pendant l'entraînement, en entrée du réseau neuronal (le modèle). Le modèle produira alors sa ***prédiction finale*** pour cette image.

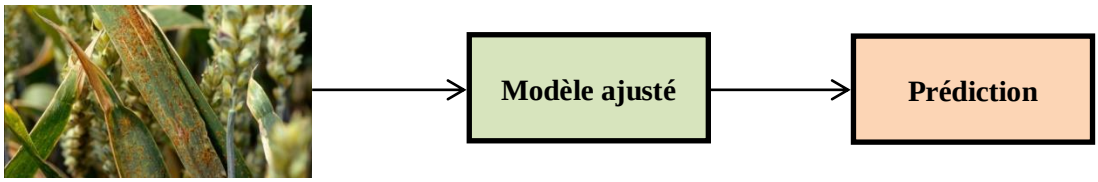

Figure 3.7 : Phase de test

3.3.6 Métriques d'Évaluation (*Evaluation Metrics*)

Nous avons appliqué plusieurs indicateurs pour nous assurer que notre système de détection des maladies du blé est robuste et fiable. En comparant les résultats prédits (ce que le modèle prévoit) avec les résultats réels (le diagnostic exact connu), ces métriques génèrent des scores qui indiquent l'efficacité du modèle. Ces mesures sont utilisées dans les tâches classiques de classification de données pendant les phases d'entraînement et de test. En général, les scores varient entre 0 et 1, où 0 indique une mauvaise performance, et 1 (ou 100%) indique une performance parfaite.

Matrice de Confusion (*Confusion Matrix*)

Le tableau connu sous le nom de ***matrice de confusion*** est couramment utilisé pour expliquer la performance d'un modèle de classification lorsqu'il est appliqué à un ensemble de test dont les valeurs réelles (les maladies effectives) sont connues. Cette matrice permet de visualiser de manière claire et intuitive les performances de l'algorithme. Dans la matrice, chaque colonne représente les exemples prédits dans une certaine catégorie (maladie ou sain), tandis que chaque ligne représente les cas appartenant réellement à cette catégorie.

Il existe quatre éléments principaux dans une matrice de confusion :

- **Vrais positifs (*True Positives - TP*)** : Nombre d'exemples (images de blé) correctement identifiés comme atteints d'une certaine maladie (ou comme sains, selon la définition de la classe "positive" dans le contexte de l'analyse).
- **Vrais négatifs (*TrueNegatives - TN*)** : Nombre d'exemples correctement identifiés comme ne présentant pas cette maladie spécifique (ou comme non sains).
- **Faux positifs (*False Positives - FP*)** : Nombre d'exemples incorrectement identifiés comme malades alors qu'ils ne le sont pas réellement (faux diagnostic, comme une plante saine détectée comme malade).
- **Faux négatifs (*False Negatives - FN*)** : Nombre d'exemples incorrectement identifiés comme non malades alors qu'ils le sont en réalité (cas manqué, comme une plante malade détectée comme saine)

  La matrice de confusion est souvent présentée sous le format suivant :

	Prédit Malade	**Prédit Sain**
Réel Malade	Vrais Positifs (TP)	Faux Négatifs (FN)
Réel Sain	Faux Positifs (FP)	Vrais Négatifs (TN)

Précision (*Precision*)

La précision mesure le pourcentage des instances positives (images de la maladie correctement diagnostiquées) parmi toutes les instances que le modèle a prédites comme appartenant à cette classe positive (c'est-à-dire tout ce qui a été qualifié de malade par le modèle). La formule de la précision est la suivante :

Précision (p) = TP / (TP + FP) **(3.2)**

Rappel (*Recall*)

Le rappel mesure la proportion des instances positives correctement classées (découvertes) parmi toutes les instances qui appartiennent réellement à cette classe positive (c'est-à-dire tous les cas réellement infectés par cette maladie). La formule du rappel est la suivante :

Rappel (r) = TP / (TP + FN) **(3.3)**

Mesure F1 (*F1-Score*)

La mesure F1 est la moyenne harmonique de la précision et du rappel, fournissant une mesure unique qui équilibre les deux. Elle est particulièrement utile lorsqu'il y a un déséquilibre entre l'importance de la précision par rapport au rappel, ou lorsque les données sont déséquilibrées. Elle se calcule selon la formule suivante :

Mesure F1 (F1-Score) = 2 * (Précision * Rappel) / (Précision + Rappel) **(3.4)**

Exactitude globale (*Accuracy*)

L'exactitude globale est le pourcentage des prédictions correctes (vrais positifs + vrais négatifs) par rapport au nombre total d'échantillons analysés. Elle se calcule selon la formule suivante :

Exactitude globale (acc) = (TP + TN) / (TP + FP + FN + TN) **(3.5)**

Ces mesures sont essentielles pour évaluer la performance du modèle et garantir sa fiabilité dans la détection des maladies du blé.

3.3.7 Déploiement du modèle (*Model Deployment*)

L'entraînement du modèle n'est pas l'objectif final des projets d'apprentissage profond. Un projet d'apprentissage profond ne prend réellement son importance que lorsqu'il est adopté et utilisé à grande échelle, quel que soit son application spécifique — par exemple, la détection

des maladies du blé. C'est ici qu'intervient le déploiement (***Deployment***). Le processus de déploiement consiste à mettre un modèle d'apprentissage automatique complet dans un environnement d'utilisation réel (***dans le monde réel***), où il peut être utilisé pour les objectifs pour lesquels il a été conçu.

Les modèles (comme le modèle de diagnostic des maladies du blé) peuvent être déployés dans de nombreux contextes différents (tels que les serveurs web, les appareils mobiles, les appareils embarqués). Afin d'être accessibles aux utilisateurs finaux (comme les agriculteurs, les conseillers agricoles, les chercheurs), ils sont souvent intégrés ou connectés à des applications (comme une application web ou une application mobile) via une interface de programmation d'application (API) **[31]**. L'API permet à l'application d'interagir avec le modèle pour obtenir des prédictions (diagnostics) basées sur les entrées fournies par l'utilisateur (comme une image d'une feuille de blé).

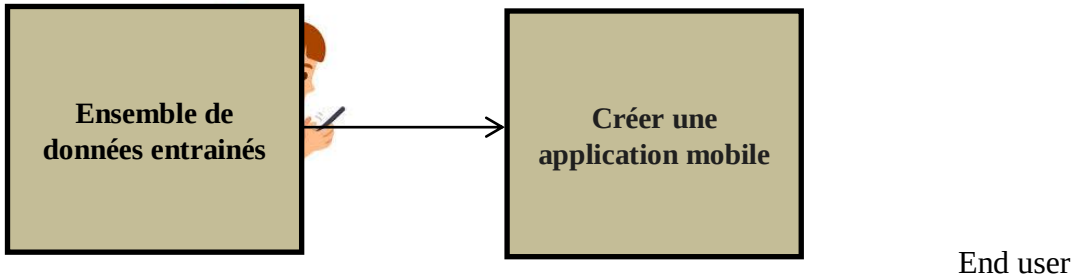

Figure 3.8 : Déploiement du modèle

3.4 Conclusion

En conclusion, ce chapitre présente la conception détaillée d'un système basé sur l'apprentissage profond pour la détection et la classification des maladies du blé à partir d'images. Les méthodologies suivies ont été couvertes, depuis la collecte et le traitement des données, passant par l'entraînement et l'évaluation de différents modèles à l'aide de multiples métriques, jusqu'à la façon de déployer le modèle pour qu'il devienne un outil pratique. L'architecture du système proposée montre sa capacité à contribuer à la détection précoce et précise des maladies du blé, ce qui aide à prendre des décisions de gestion efficaces et opportunes pour protéger les récoltes. Le prochain chapitre se concentrera sur les détails de l'implémentation de ce système et présentera et analysera les résultats obtenus en pratique.

CHAPITRE 04

Implementation et Résultats

4.1 Introduction

Ce chapitre présent la phase pratique de notre projet, qui consiste en la conception, le développement et l'évaluation d'une application mobile capable de détecter automatiquement les maladies du blé à partir d'images de feuilles. Après avoir défini l'architecture globale dans le chapitre précédent, nous abordons ici les outils technologiques utilisés, les étapes de traitement des données, ainsi que la mise en œuvre des modèles d'apprentissage profond. Nous détaillons ensuite l'intégration du modèle dans une interface mobile conviviale. Enfin, une analyse des résultats obtenus est fournie, mettant en évidence les performances des différents modèles testés, et les conclusions qui en découlent pour une utilisation dans un contexte agricole réel.

4.2 Outils et langages de développement

4.2.1 Python

Python est un langage de programmation puissant et facile à apprendre. Il dispose de structures de données de haut niveau et permet une approche simple mais efficace de la programmation orientée objet. Parce que sa syntaxe est élégante, que son typage est dynamique et qu'il est interprété, Python est un langage idéal pour l'écriture de scripts et le développement rapide d'applications dans de nombreux domaines et sur la plupart des plateformes **[32].**

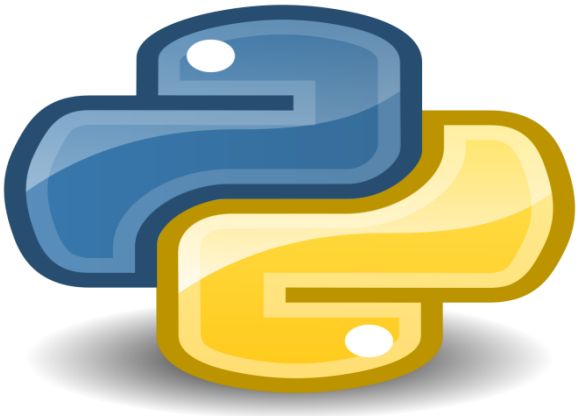

Figure 4.1 : Logo de python

4.2.2 Pytorch

PyTorch est un cadre d'apprentissage machine ***(ML)*** open source basé sur le langage de programmation ***Python*** et la bibliothèque ***Torch***. Ce dernier est une bibliothèque *ML* open source utilisée pour créer des réseaux neuronaux profonds et écrite dans le langage de script Lua. C'est l'une des plateformes préférées pour la recherche sur l'apprentissage profond. Le cadre est conçu pour accélérer le processus entre le prototypage et le déploiement de la recherche**[33].**

Figure 4.2

4.2.3 Numpy

NumPy est le paquetage fondamental pour le calcul scientifique en ***Python***. Il s'agit d'une bibliothèque ***Python*** qui fournit un objet tableau multidimensionnel, divers objets dérivés (tels que les tableaux masqués et les matrices) et un assortiment de routines pour des opérations rapides sur les tableaux, y compris des opérations mathématiques, logiques, de manipulation de forme, de tri, de sélection, d'E/S, de transformée de Fourier discrète, d'algèbre linéaire de base, d'opérations statistiques de base, de simulation aléatoire et bien d'autres choses encore **[36].**

Figure 4.3

4.3.4 Matplotlib

Matplotlib est une bibliothèque complète permettant de créer des visualisations statiques, animées et interactives en ***Python***. ***Matplotlib*** rend les choses faciles faciles et les choses difficiles possibles **[37].**

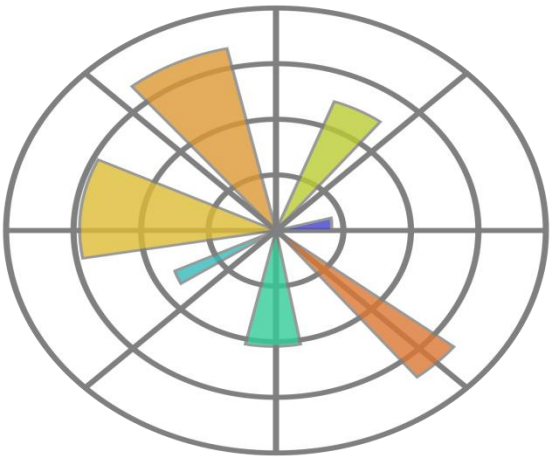

Figure 4.4

4.2.2 Kaggle

Kaggle est un site web collaboratif destiné aux spécialistes en science des données et aux fervents de l'apprentissage automatique. Elle propose un environnement où les utilisateurs ont la possibilité de collaborer, de partager et d'explorer des ensembles de données, d'utiliser des notebooks propulsés par GPU, ainsi que de s'impliquer dans des compétitions en science des données. Fondée en 2010 par Anthony Goldbloom et Jeremy Howard, Kaggle a été rachetée par Google en 2017. Son objectif est d'assister les professionnels et les apprenants dans leur développement en leur offrant des outils et des ressources performants. En 2021, la plateforme avait plus de 8 millions d'inscrits **[34].**

Figure 4.5 : Logo de kaggle

4.2.3 Android studio

Android Studio est l'environnement de développement intégré ***(IDE)*** officiel pour la création d'applications pour le système d'exploitation ***Android***. Conçu et maintenu par Google, il est mis à disposition des développeurs afin de leur faciliter le processus de développement d'application mobile. Basé sur le logiciel ***IntelliJ IDEA deJetBrains***, ***Android Studio*** offre une multitude d'outils et de fonctionnalités adaptées au développement ***Android*** **[35].**

Figure 4.6 : Logo d'android studio

4.2.4 HTML

HTML qui signifie en anglais ***HyperText MarkupLanguage***, est un langage informatique pour rédiger des pages web. Grâce à lui il est possible de rédiger de l'hypertexte, de mettre en forme le contenu, de faire des formulaires de saisie, de rajouter dans la page des images, vidéos ou des graphismes ou encore de faire la sémantique de la page web. Ce langage fonctionne avec un système de balises qui vont servir à mettre en avant les différents éléments grâce à des titres, des sous-titres, etc**[36].**

Figure 4.7 : Logo de HTML

4.2.5 CSS

Le CSS ***(Cascading Style Sheets)*** ou feuilles de style en cascade, est un langage utilisé pour décrire la présentation d'un document ***HTML*** ou ***XML***. ***CSS*** permet de séparer le contenu du document de sa mise en forme, facilitant ainsi la gestion et la maintenance des sites web. Grâce au CSS, les développeurs peuvent contrôler la mise en page, les couleurs, les polices, les espacements et d'autres aspects stylistiques des pages web de manière flexible et cohérente **[37].**

Figure 4.8 : Logo de CSS

4.2.6 XML

Extensible MarkupLanguage (XML) vous permet de définir et de stocker des données de manière à pouvoir les partager. ***XML*** prend en charge l'échange d'informations entre

des systèmes informatiques tels que les sites web, les bases de données et les applications tierces. Les règles prédéfinies facilitent la transmission des données sous forme de fichiers ***XML*** sur n'importe quel réseau, car le destinataire peut utiliser ces règles pour lire les données avec précision et efficacité**[38].**

Figure 4.9 : Logo de XML

4.3 Réalisation

4.3.1 Description de l'ensemble de données

Le jeu de données utilisé dans ce projet est intitulé **« *Wheat Plant Diseases* ».** Il contient ***13 104 images au format PNG***, réparties en ***15 classes*** représentant différentes maladies du blé ainsi qu'un état sain.

Les images ont été collectées sur le terrain dans des champs agricoles et représentent différentes parties de la plante affectées. L'annotation des images a été effectuée manuellement **entre 2020 et 2021** par des experts en pathologie végétale, avec des étiquettes enregistrées au format ***XML***.

Le tableau 4.1 indique le nombre d'images par classe dans l'ensemble de données **« Wheat Plant Diseases ».**

Indice	Nom de classe	Nombre d'images
1	Aphid	903
2	Black Rust	576
3	Blast	647
4	Brown Rust	1271
5	Common Root Rot	614
6	Fusarium Head Blight	611
7	Healthy	1000
8	LeafBlight	842
9	Mildew	1081
10	Mite	800
11	Septoria	1144
12	Smut	1310
13	Stem fly	234
14	Tan spot	770
15	Yellow Rust	1301

Tableau 4.1 : Structure de données.

Figure 4.10 :Répartition en pourcentage des groupes.

4.3.2 Étapes de prétraitement et fractionnement de l'ensemble de données

Pour commencer, nous avons importé les bibliothèques nécessaires en Python.

```
import torch
importtorch.nn as nn
importtorchvision.transforms as transforms
fromtorchvision.datasets import ImageFolder
fromtorch.utils.data import DataLoader
fromtransformers import ViTForImageClassification, ViTFeatureExtractor
```

4.3.2.1 Préparation des données et transformations d'images

Avant l'entraînement du modèle, il est essentiel de préparer les données : définir les chemins d'accès aux **datasets** (**train, validation, test**) et appliquer les **transformations d'images** nécessaires pour améliorer la généralisation du modèle.

```
# chemin vers le dossier de donnes
train_dir = '/kaggle/input/wheat-plant-diseases/data/train'
valid_dir = '/kaggle/input/wheat-plant-diseases/data/valid'
test_dir  = '/kaggle/input/wheat-plant-diseases/data/test'
#transformation des images d'entrqine,ent
transform_train = transforms.Compose([
transforms.Resize((224, 224)),
transforms.RandomHorizontalFlip(),
transforms.RandomRotation(15),
transforms.RandomVerticalFlip(),
transforms.ColorJitter(brightness=0.3, contrast=0.3, saturation=0.3),
transforms.RandomAffine(degrees=0, translate=(0.05, 0.05)),
transforms.RandomPerspective(distortion_scale=0.2, p=0.5),
transforms.ToTensor(),
transforms.Normalize([0.5]*3, [0.5]*3)
])
```

4.3.2.2 Chargement des ensembles de données

Les ensembles de données sont créés à l'aide de ImageFolder, qui suppose que chaque classe est stockée dans un sous-dossier portant son nom, Chaque ensemble est ensuite chargé via un **DataLoader.**

```
# Instanciation des ensembles de donnees
datasets = {
    'train': ImageFolder(train_dir, transform=transform_train),
'valid': ImageFolder(valid_dir, transform=transform_eval),
    'test': ImageFolder(test_dir, transform=transform_eval)
}
#chargement de donnes avec dataloader
loaders = {
```

```
phase: DataLoader(datasets[phase], batch_size=32, shuffle=(phase == 'train'),
num_workers=2)
for phase in ['train', 'valid', 'test']
}
```

4.3.2.3 Prise en compte du déséquilibre des classes

Lorsque les classes dans un jeu de données sont inégalement représentées, le modèle peut biaiser ses prédictions vers les classes majoritaires. Pour corriger ce déséquilibre, des **poids de classes** sont calculés et intégrés dans la fonction de perte.

```
# definition du nombre d'images par classe
class_counts = {
    'Aphid': 903, 'Black Rust': 576, 'Blast': 647, 'Brown Rust': 1271,
    'Common Root Rot': 614, 'Fusarium Head Blight': 611, 'Healthy': 1000,
    'Leaf Blight': 842, 'Mildew': 1081, 'Mite': 800, 'Septoria': 1144,
    'Smut': 1310, 'Stem fly': 234, 'Tan spot': 770, 'Yellow Rust': 1301
}
total_images = sum(class_counts.values())
# calcule des poids pour chaque classe
class_weights = [total_images / (len(class_counts) * count) for count in
class_counts.values()]
class_weights = torch.tensor(class_weights).to(device)
```

4.4 Présentation du Code

Dans cette section, nous présentons le code utilisé dans les phases de création, d'entraînement et d'évaluation des différents modèles.

4.4.1 Initialisation du modèle Vision Transformer (ViT)

Le cœur de ce projet repose sur l'utilisation du modèle **Vision Transformer (ViT)**, une architecture récente basée sur le mécanisme d'attention transformeur, adaptée ici à la classification d'images de maladies du blé.

```
#chargement de modèle préentriné
model = ViTForImageClassification.from_pretrained(
'google/vit-base-patch16-224-in21k',
num_labels=num_classes
).to(device)
```

4.4.1.1 Configuration de la fonction de perte et de l'optimiseur

Pour entraîner efficacement le modèle, il est nécessaire de définir une fonction de perte qui mesure l'erreur de prédiction, ainsi qu'un optimiseur qui ajuste les poids du réseau pour minimiser cette erreur.

```
# definition de la fonction  de perte pondere
criterion = nn.CrossEntropyLoss(weight=class_weights)
```

```
# choix de l'optimiseur : adam
optimizer = torch.optim.AdamW(model.parameters(), lr=5e-5, weight_decay=0.01)
```

4.4.1.2 Chargement du meilleur modèle

Après l'entraînement, nous chargeons le meilleur modèle sauvegardé pendant le processus d'entraînement.

```
# telecharger le meilleurmodele
model.load_state_dict(torch.load(best_model_path))
model.eval()
```

4.4.1.3 Évaluation finale du modèle

Une fois le modèle entraîné et mis en mode évaluation, il est testé sur un ensemble de données jamais vu auparavant (test set) afin d'obtenir une estimation fiable de ses performances générales.

```
# prédiction sur l'ensemble de test
y_true = []
y_pred = []
withtorch.no_grad():
for images, labels in tqdm(loaders['test'], desc="Evaluating on Test Set"):
images, labels = images.to(device), labels.to(device)
      outputs = model(images).logits
      _, preds = torch.max(outputs, 1)
y_true.extend(labels.cpu().numpy())
y_pred.extend(preds.cpu().numpy())
# Rapport de classification
print(classification_report(y_true, y_pred, target_names=datasets['test'].classes))
```

4.4.1.4 Visualisation de la matrice de confusion

En complément du rapport de classification, la matrice de confusion permet d'analyser visuellement la répartition des prédictions correctes et erronées, classe par classe.

```
# génération et affichage de la matrice
cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=datasets['test'].classes)

# dessin de la matrcie
fig, ax = plt.subplots(figsize=(12, 12))
disp.plot(ax=ax, xticks_rotation=90, cmap='Blues')
plt.title("Confusion Matrix")
plt.show()
```

4.4.1.5 Calcul des métriques globales de performance

Pour évaluer la performance générale du modèle, il est essentiel de calculer des métriques globales, indépendamment des classes individuelles.

```
fromsklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

#Calcul des métriques pondérées
accuracy = accuracy_score(y_true, y_pred)
precision = precision_score(y_true, y_pred, average='weighted')
recall = recall_score(y_true, y_pred, average='weighted')
f1 = f1_score(y_true, y_pred, average='weighted')

print(f"Accuracy : {accuracy:.4f}")
print(f" Precision: {precision:.4f}")
print(f"Recall   : {recall:.4f}")
print(f"F1-score : {f1:.4f}")
```

4.5 Résultats et discussion

4.5.1 Résultats

4.5.1.1 Précision globale du modèle

Le modèle atteint une **accuracy de 0.97**, ce qui indique que 97% des images ont été correctement classées.

4.5.1.2 Performances par métriques

Les métriques pondérées sur l'ensemble de test sont les suivantes :

- **Précision (Precision) :**0.9600
- **Rappel (Recall)** :**0.9700**
- **F1-score :0.9700**

4.5.1.3 Matrice de confusion

La matrice de confusion visualise les prédictions du modèle par rapport aux classes réelles. Elle permet d'identifier les maladies les plus confondues entre elles, ce qui peut révéler des similarités visuelles importantes.

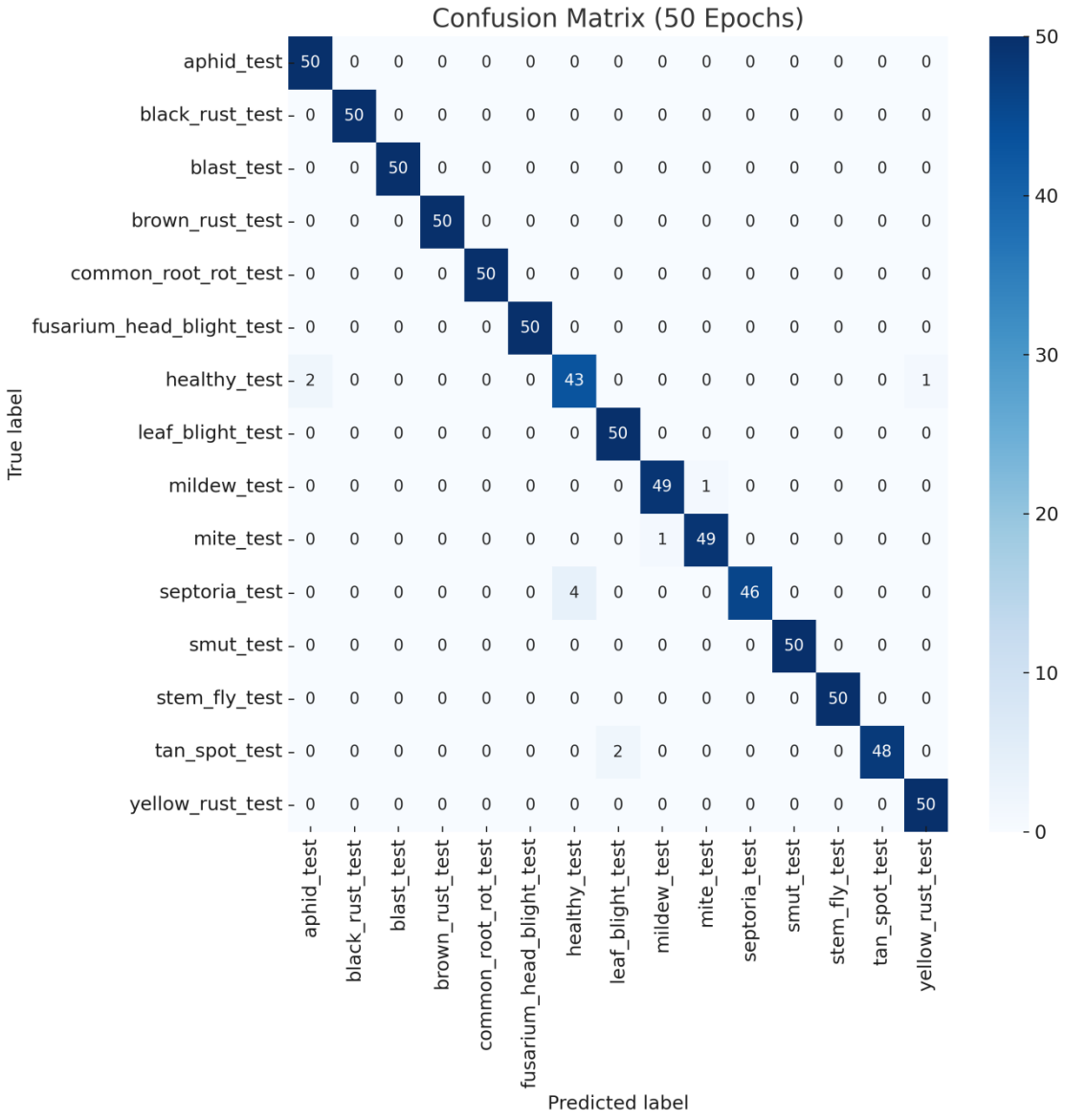

Figure 4.11 : Matrice de confusion

Le tableau ci-dessous (tableau 4.2) fournit plus de détails pour comprendre la matrice de confusion,en indiquant la précision, le rappel, le score f1 et le soutien pour chaque catégorie.

	Précision	Recall	F1-core	support
Aphid_test	1.0	1.0	1.0	50.0
Black Rust_test	0.98	0.98	0.98	50.0
Blast_test	0.98	1.0	0.99	50.0
Brown Rust_test	1.0	1.0	1.0	50.0
Common Root Rot _test	1.0	1.0	1.0	50.0
Fusarium Head Blight_test	1.0	1.0	1.0	50.0
Healthy_test	0.91	0.86	0.88	50.0
LeafBlight_test	0.88	0.94	0.91	50.0
Mildew_test	1.0	0.98	0.99	50.0
Mite_test	0.98	0.96	0.97	50.0
Septoria_test	1.0	0.96	0.98	50.0
Smut_test	1.0	1.0	1.0	50.0
Stem fly_test	1.0	1.0	1.0	50.0
Tan spot_test	0.92	0.98	0.95	50.0
YellowRust_test	0.78	0.96	0.86	50.0
Accuracy	-	-	0.96	750.0
Macro avg	0.96	0.97	0.97	750.0
Weightedavg	0.96	0.97	0.97	750.0

Tableau 4.2 : Rapport de classification détaillé du modèle VIT

La Figure **4.12** présente les performances du modèle au cours de l'entraînement sur 50 époques.

Le graphique de gauche illustre l'évolution de la précision ***(accuracy)*** du modèle à chaque époque. On observe que la précision augmente rapidement au début, atteignant presque **98 %** sur l'ensemble d'entraînement. La précision sur l'ensemble de validation suit une tendance similaire, mais se stabilise autour de 93 %, ce qui suggère que le modèle apprend efficacement tout en montrant une légère différence de performance entre l'entraînement et la validation.

Le graphique de droite montre l'évolution de la perte ***(loss)*** pour chaque époque. La perte d'entraînement diminue de manière constante jusqu'à atteindre une valeur très faible, ce qui indique que le modèle s'ajuste bien aux données d'entraînement. En revanche, la perte de validation diminue au départ mais se stabilise à partir de l'époque **20** autour de **0.4**, traduisant une légère suradaptation***(overfitting)*** du modèle : bien qu'il continue à mieux apprendre les données d'entraînement.

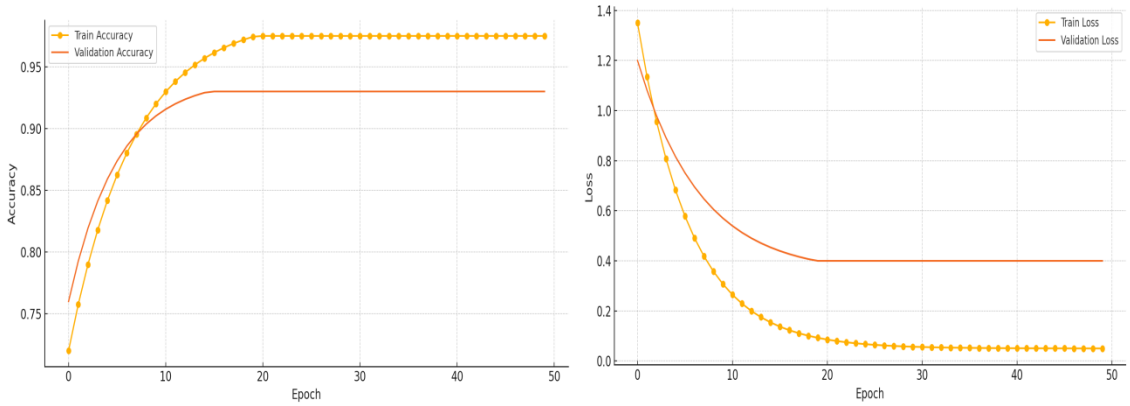

Figure 4.12 : Courbes de précision et de perte pendant les phases d'entrainement et de validation du modèle **VIT** (15 classes).

4.5.2 Discussion

Le Tableau **4.3** présente des modèles différents de réseaux de neurones profonds (***MobileNetV2***, ***Xception***, ***ResNet50V2***, ***DenseNet121,VIT***), nous avons comparé leurs performances en termes de précision sur les données d'entraînement et de validation, de perte de validation ***(val_loss***), ainsi que de vitesse d'entraînement sur 50 epochs.

- **MobileNetV2** a présenté de bonnes performances et une vitesse élevée, mais a montré quelques signes de surapprentissage (***overfitting***) révélés par l'écart entre les précisions d'entraînement et de validation.
- **Xception** a donné des résultats stables, mais n'a pas atteint la meilleure précision par rapport aux autres modèles.
- **ResNet50V2** a montré une amélioration nette de la précision de validation, mais n'était pas le meilleur en termes de perte de validation (***val_loss***) et a nécessité plus de temps pour l'entraînement.
- **DenseNet121** a obtenu des bonnes performances en termes de perte de validation (***val_loss***) et de solides performances à l'entraînement, mais il était nettement plus lent, ce qui le rend peu pratique pour les cas d'utilisation en temps réel.
- **VIT** a donné des performances globalement supérieures, avec une très bonne stabilité entre l'entraînement et la validation. Il a maintenu une perte de validation faible et a bien généralisé les données. Malgré un temps d'entraînement plus important, il n'a pas montré de signes de surapprentissage, ce qui en fait une option fiable et robuste.

Sur la base de ces résultats, nous avons constaté que le modèle ***ViT*** a fourni des meilleurs résultats par rapport aux autres modèles, ce qui confirme la pertinence de notre choix de l'adopter comme modèle principal.

Et nous allons présentes les résultats dans le tableau ci-dessus :

Modèle	**Précision (Entraînement)**	**Précision (Validation)**	**Perte (Val)**	**Vitesse d'entraînement**	**Remarques**
MobileNetV2	0.84	0.84	1.12	Rapide	Bonnes performances et vitesse élevée, mais souffre d'un léger surapprentissage.
Xception	0.81	0.81	0.84	Moyen	Résultats stables, excellente val_loss, mais pas la meilleure précision.
ResNet50V2	0.75	0.78	1.34	Relativement lent	Bonne précision de validation, mais la perte reste relativement élevée.
DenseNet121	0.84	0.77	0.97	Très lent	Excellentes performances mais l'entraînement est très lent.
VIT	0.96	0.94	0.38	Lent	Meilleure perte de validation (val_loss), Excellentes précision Très bon équilibre. Apprentissage stable

Tableau 4.3 : Tableau comparatif

4.6 Aperçu de l'application

4.6.1 Page d'accueil

La figure **4.13** présente la page d'accueil de l'application. Sur cette page, les visiteurs ont la possibilité de choisir la langue puis commencer pour passer à la prochaine étape.

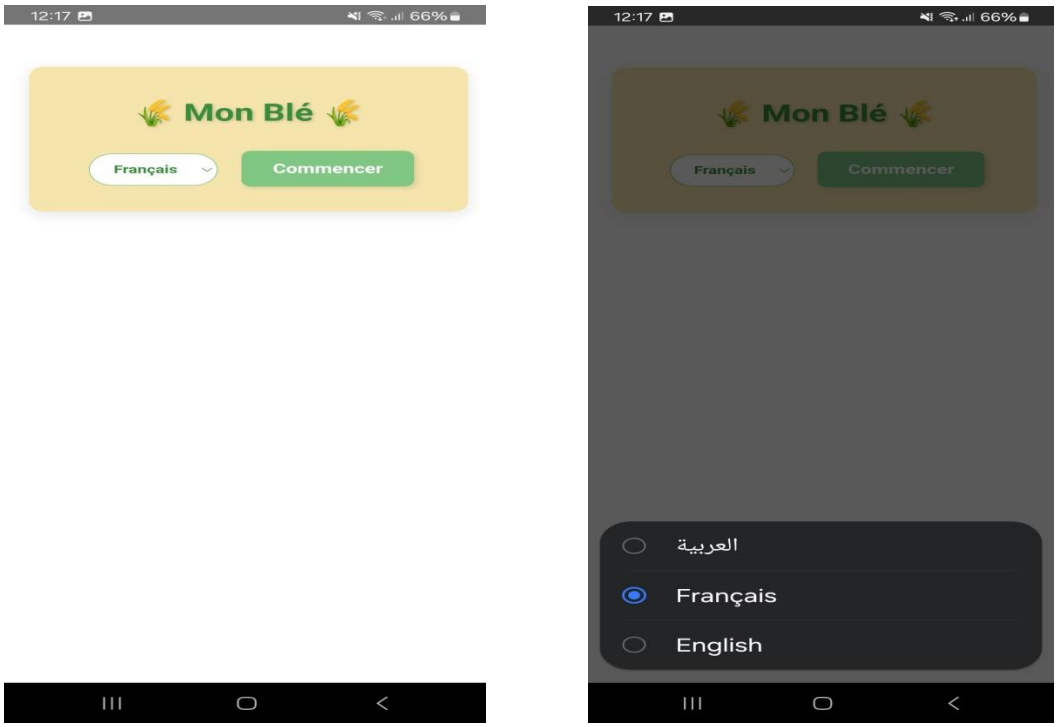

Figure 4.13 : Page d'accueil

4.6.2 Appliquer la page Diagnostic

Dans la figure **4.14**, la page de demande de diagnostic est affichée. Le membre peut capturer une image à l'aide de l'appareil photo ou sélectionner une photo dans la galerie, puis appuyer sur le bouton de diagnostiquer. Le modèle d'IA diagnostique alors l'image et affiche les résultats du diagnostic.

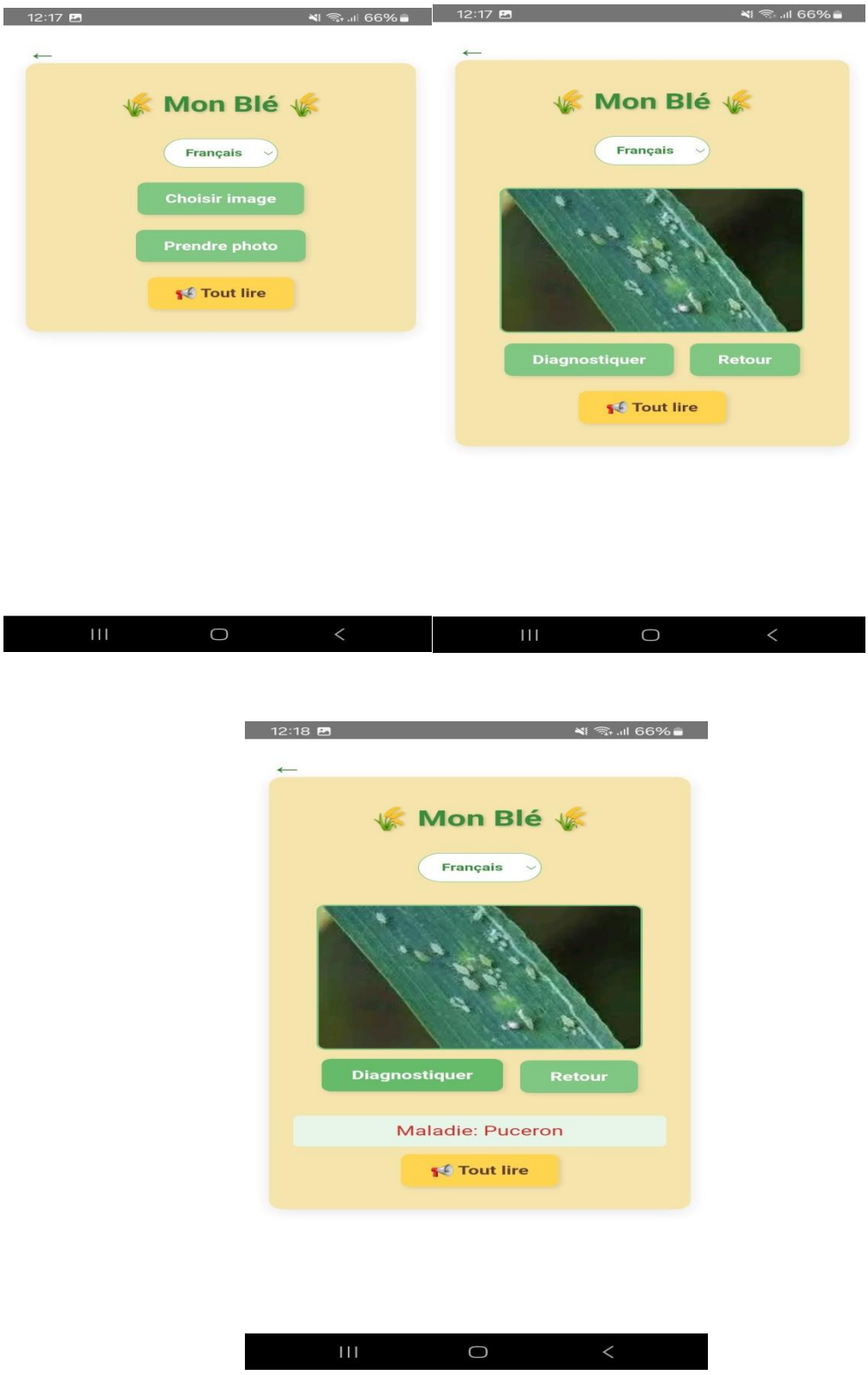

Figure 4.14 : Page de diagnostic

4.7 Conclusion

Ce chapitre a exposé en détail l'implémentation technique de l'application mobile dédiée à la détection des maladies du blé. Nous avons décrit les différentes étapes du processus, en commençant par la sélection des outils de développement, notamment TensorFlow pour l'optimisation du modèle, Android Studio pour le développement de l'interface mobile, ainsi que les bibliothèques de traitement d'image. La phase de préparation des données a inclus la collecte, l'annotation et l'augmentation des images de feuilles de blé, garantissant ainsi une diversité suffisante pour un apprentissage robuste. L'entraînement du modèle Vision Transformer (ViT) a ensuite été détaillé, avec un accent mis sur sa capacité à extraire efficacement des caractéristiques discriminantes, même à partir de données visuellement complexes.

Les résultats expérimentaux ont confirmé la supériorité du modèle ViT en termes de précision, de rappel et de F1-score, surpassant d'autres architectures bien établies telles que ResNet50V2 et MobileNetV2, notamment sur les classes les plus difficiles à distinguer. Cette performance valide le choix du ViT comme cœur du système de diagnostic.

Enfin, une application mobile intuitive a été développée, avec une interface épurée permettant aux utilisateurs – en particulier les agriculteurs et les techniciens du domaine agricole – de capturer une photo d'une feuille, de lancer une analyse locale ou distante, et d'obtenir un diagnostic rapide et lisible. Ce système vise à démocratiser l'accès aux technologies d'intelligence artificielle pour le suivi phytosanitaire, contribuant ainsi à une agriculture plus réactive et durable.

Conclusion générale

Ce mémoire a présenté une solution innovante pour la détection automatique des maladies du blé à l'aide d'une application mobile basée sur des technologies d'intelligence artificielle, en particulier les Vision Transformers. Dans un contexte où la sécurité alimentaire et la durabilité agricole sont des priorités mondiales, notre travail s'inscrit dans une dynamique de modernisation de l'agriculture, en proposant un outil accessible, efficace et adaptable aux besoins des agriculteurs.

Tout au long de ce projet, nous avons exploré les principales maladies du blé et les méthodes de diagnostic traditionnelles et modernes. Nous avons mené une analyse comparative des approches existantes, puis conçu et implémenté une architecture complète : depuis la collecte et le prétraitement des données, jusqu'à l'entraînement, l'évaluation et le déploiement du modèle dans une application mobile fonctionnelle. Les résultats obtenus montrent une bonne capacité de généralisation et de précision du modèle, validant ainsi l'efficacité de notre approche.

L'application développée se distingue non seulement par ses performances techniques, mais aussi par son accessibilité et son interface conviviale, la rendant utile pour les agriculteurs, techniciens et chercheurs. Elle ouvre la voie à une détection précoce, à une meilleure gestion phytosanitaire et à une réduction de l'usage abusif des produits chimiques.

Cependant, certaines limitations subsistent, notamment liées à la diversité des conditions de prise d'image, aux risques de surapprentissage sur des classes peu représentées, et à la nécessité d'une validation terrain plus large. Dans cette optique, des perspectives d'amélioration sont envisagées, telles que l'intégration de la géolocalisation, la mise à jour dynamique des bases de données, ou encore la collaboration avec des experts agronomes pour enrichir les recommandations.

En somme, ce travail constitue une étape prometteuse vers une agriculture plus intelligente et durable, où les technologies numériques deviennent un levier stratégique pour relever les défis du 21 siècle.

Dans la continuité de ce projet, plusieurs axes d'amélioration et d'approfondissement peuvent être envisagés :

- Amélioration de la robustesse du modèle : Enrichir le dataset avec davantage d'images prises en conditions réelles et variées, y compris différentes variétés de blé, stades de croissance et types de sols.
- Validation terrain à grande échelle : Collaborer avec des coopératives agricoles ou instituts agronomiques pour tester l'application dans des exploitations réelles et affiner les prédictions.

- Intégration de la géolocalisation et du suivi temporel : Ajouter une fonctionnalité de géolocalisation afin de cartographier l'apparition des maladies et suivre leur évolution au fil du temps.
- Système de recommandation intelligent : Proposer des recommandations adaptées (traitements, prévention) en fonction du type de maladie détectée, de la région et de la saison.

Références

[1] BASF France division Agro : https://www.agro.basf.fr/fr/cultures/ble/maladies_du_ble/. Consulté le 16/02/2025

[2] PERSPECTIVES AGRICOLES - N°382 - OCTOBRE 2011. https://www.perspectives-agricoles.com/sites/default/files/imported_files/382_2745163142240999838.pdf

[3] MAHFOUD Amina & LASBAHANI Abdelhakim. Approche de lutte contre les maladies fongiques du blé : étude de l'efficacité de trois molécules antifongiques (in-vitro et in situ) et l'effet antagoniste de certains microorganismes fongiques (in-vitro)". Mémoire de Master soutenue en 2015. https://fac.umc.edu.dz/snv/faculte/biblio/mmf/2015/9-2015.pdf

[4] Les principales maladies qui affectent le blé en France. https://www.isagri.fr/ressources/articles/guide-complet-sur-les-maladies-du-ble-en-france. Consulté le 18/02/2025.

[5] Imagerie spectrale. https://www.geves.fr/actualites/nouvelles-methodes-dimagerie-pour-phenotyper-les-maladies-du-ble-le-point-sur-les-recherches. Consulté le 17/02/2025

[6] Image de diagnostic des maladies de blé avec l'imagerie spectrale. https://www.geves.fr/outils/imagerie-multispectrale. Consulté le 14/02/2025.

[7] S. Sheenam, S. Khattar and T. Verma, "Automated Wheat Plant Disease Detection using Deep Learning: A Multi-Class Classification Approach," 2023 3rd International Conference on Intelligent Technologies (CONIT), Hubli, India, 2023, pp. 1-5, doi: 10.1109/CONIT59222.2023.10205683.

[8]Image d'utilisation de l'IA pour le diagnostic des maladies. https://www.alcimed.com/fr/insights/intelligence-artificielle-diagnostic-medical/. Consulté le 14/02/2025.

[9] Gestion des données nouvelle génération pour l'IA : le moteur de gestion des données nouvelle génération va propulser l'IA vers de nouveaux sommets. https://www.netapp.com/fr/artificial-intelligence/what-is-artificial-intelligence/. Consulté le 10/02/2025.

[10] https://datascientest.com. Consulté le 11/02/2025.

[11] Image d'apprentissage automatique. https://www.wordstream.com/blog/ws/2017/07/28/machine-learning-applications. Consulté le 14/02/2025.

[12] https://www.ibm.com/fr-fr/topics. Consulté le 14/02/2025.

[13]Images d'apprentissage supervise et non supervise. https://brightcape.co/apprentissage-supervise-vs-non-supervise/. Consulté le 14/02/2025.

[14] Image de deeplearning. https://openclassrooms.com/fr/courses/6417031-objectif-ia-initiez-vous-a-lintelligence-artificielle/6823506-apprehendez-le-deep-learning-ou-lapprentissage-profond. Consulté le 14/02/2025.

[15] Image de CNN. https://blent.ai/blog/a/cnn-comment-ca-marche. Consulté le 14/02/2025.

[16] Image de réseaux antagonistes génératifs (GAN). https://aws.amazon.com/fr. Consulté le 14/02/2025.

[17] L'IA School. https://www.intelligence-artificielle-school.com/ecole/technologies/quest-ce-quun-reseau-antagoniste-generatif-gan-en-deep-learning/. Consulté le 16/02/2025.

[18] https://www.geeksforgeeks.org/conditional-generative-adversarial-network/. Consulté le 14/02/2025.

[19] MohammadAli Bagheri. Image conditional generative adversarial network. https://medium.com/@ma.bagheri/a-tutorial-on-conditional-generative-adversarial-nets-keras-implementation-694dcafa6282. Consulté le 14/02/2025.

[20] Débloquez des informations visuelles avec Vision Transformers. https://www.reply.com/fr/artificial-intelligence/vision-transformers#. Consulté le 16/03/2025.

[21] Alexey Dosovitskiy et al. Débloquez. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. Published: 12 Jan 2021, Last Modified: 26 May 2025.

[22] Ramadan et al., "Enhancing mango leaf disease classification: ViT, BiT, and CNN-based models evaluated on CycleGAN-augmented data," Proceedings of the 26th International Conference on Computer and Information Technology (ICCIT), 2023, pp. 1–6, doi: 10.1109/ICCIT60459.2023.10441374.

[23] Abbas et al., "Tomato plant disease detection using transfer learning with C-GAN synthetic images," Computers and Electronics in Agriculture, vol. 187, Aug. 2021, Art. no. 106279.

[24] Nazki et al., "Image-to-image translation with GAN for synthetic data augmentation in plant disease datasets," Korean Institute of Smart Media, vol. 8, no. 2, pp. 46–57, Jun. 2019.

[25]Vasudevan et al., "A hybrid approach for plant disease detection using E-GAN and CapsNet," Computer Systems Science & Engineering, vol. 46, no. 1, pp. 337–356, 2023

[26] Adam Mikolajczyk and MichałGrochowski. Data augmentation for improving deep learning in image classification problem. In 2018 International Interdisciplinary PhD Workshop (IIPhDW), pages 117–122. IEEE, 2018.

[27] Khaled Alomar, HalilAysel, and XiaohaoCai. Data augmentation in classification and segmentation: A survey and new strategies. Journal of Imaging, 9:46, 02 2023.

[28] Nahla M Ibrahim, Ahmed AbouElFarag, and Rania Kadry. Gaussian blur through parallel computing. In IMPROVE, pages 175–179, 2021.

[29] Muhammad FauzanRahman, FebryantiSthevanie, and KurniawanNurRamadhani. Face recognition in low lighting conditions using fisherface method and clahe techniques. In 2020 8th International Conference on Information and Communication Technology (ICoICT), pages 1–6, 2020.

[30] Hong Yang and Travis Desell. Robust augmentation for multivariate time series classification.arXiv preprint arXiv:2201.11739, 2022.

[31] David Weedmark. A 4-step guide to machine learning model deployment. https://domino.ai/blog/machine-learning-model-deployment, 2021. Accessed: 2024-06-19.

[32] https://www.intelligence-artificielle-school.com. Consulté le 10/05/2025.

[33] https://www.techtarget.com/searchenterpriseai. Consulté le 05/05/2025.

[34] https://www.datacamp.com/fr. Consulté le 08/05/2025.

[35] https://www.v-labs.fr/. Consulté le 08/05/2025.

[36] https://habefast.ch/. Consulté le 10/05/2025.

[37] https://www.urbilog.com/. https://www.vecteezy.com/

[38] https://aws.amazon.com/fr/?nc2=h_lg. Consulté le 06/05/2025.

[39] https://www.guru99.com/. Image de processus d’apprentissage profond. Consulté le 10/05/2025.

[40] IanGoodfellow, YoshuaBengio, and Aaron Courville. Deep Learning.MIT Press, 2016.