Democratic and Popular Republic of Algeria
Ministry of Higher Education and Scientific Research
University of Mohamed Khider - BISKRA
Faculty of Exact Sciences
Computer Science Department

Order N^{o} : IA_Start Up_02/M2/2025

Master Dissertation

Submitted in fulfillment of the requirements for the Master's degree in

Computer Science

Option: **Artificial Intelligence (AI)**

Leveraging NLP for Plant Disease Detection in Smart Farms

Presented by:

Sara BITAM

Graduating on 17/06/2025 in front of the following committee of juries: :

Adel ABDELLI	**MCB**	**Examiner**
Sihem SLATNIA	**Professor**	**President**
Abdelhak MERIZIG	**MCA**	**Supervisor**
Imane YOUKANA	**MCB**	**Co-Supervisor**

Session 2024/2025

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Dedication

بِسْمِ اللهِ الرَحمَنِ الرَحيمِ

وَفَوْقَ كُلِ ذِي عِلْمٍ عَلِيمٌ

الْحَمْدُ لِلَه الَذي مَنَحَنِي الْقُوَة وَالْحِكْمَةَ وَالصَبرَ لِإِتْمَامِ هَذَا الْعَمَلِ.
وَلَوْلَا رَحْمَتُهُ الْوَاسِعَةُ وَهُدَاهُ، لَمَا كَانَ لِهَذَا الإِنْجَازِ أَنْ يَتَحَقَق.

إلى والديَ الحبيبين، **محمد مهدي بيطام** و **سهام بن ساهل** ،
على حبكما غير المشروط، ودعمكما الدائم، وتضحياتكما التي لا تُعد ولا تُحصى...

إلى إخوتي، **رائد** و **خليل** ،
رفقتكما وتشجيعكما كانا مصدر قوة دائمة لي...

إلى ذكرى أخي **محسن** وجدتي **ميمي** الغالية،
وجودكما لا يزال حيًّا في قلبي، يُلهمني في كل خطوة أخطوها...

إلى صديقتي العزيزتين، **أميرة** و **مروى** ،
شكرًا لمشاركتكما هذه الرحلة الأكاديمية معي، بالضحك والصبر...

أُهدي هذا العمل إليكم جميعًا بكل حب وامتنان.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, **Dr. Abdelhak MERIZIG** and **Dr. Imane YOUKANA**, for their insightful guidance, continuous encouragement, and unwavering support throughout the course of this project. I am especially thankful to **Professor Laid KAHLOUL**, who was my supervisor and continues to provide invaluable guidance, for generously sharing his knowledge and always being available to help.

This project would not have been possible without the valuable expertise of **Rabie BENSAHEL** and **Aicha SOLTANE**, agricultural engineers whose profound knowledge in the field was instrumental in shaping the foundation of this work. I also extend my heartfelt thanks to **Tahraoui Company**, particularly **Nadina TAHRAOUI**, for her significant contribution to the survey component.

I am immensely grateful to my uncle and mentor, **Salim BITAM**, for his steadfast guidance and teaching throughout my academic journey. I also thank the **LINFI** and **COMMAND Laboratories** for providing inspiring environments that foster research and innovation. My deep appreciation goes to **Abdelhakim DRID**, a PhD student who generously allowed me to use his laptop for training my models, which was crucial for the technical implementation of this project. I also extend my gratitude to **Fella ACHOUR** for her valuable contributions to the startup and entrepreneurial aspects.

Finally, I extend my heartfelt thanks to all the teachers who have accompanied me on my educational path. Your knowledge, patience, and encouragement have shaped both my intellect and character.

With sincere gratitude,

Sara BITAM

Abstract

Plant diseases compromise global crop production by **20–40% annually**, threatening agricultural sustainability and food security. In developing regions, farmers frequently lack timely access to phytopathological expertise and advanced diagnostic tools for effective disease management. This study presents an integrated AI-powered system for plant disease detection that combines **visual analysis** and **treatment recommendation** capabilities to support both individual farmers and agricultural enterprises.

The system architecture implements a cascading multi-model approach: a **MobileNetV2-based out-of-distribution (OOD) detection module** validates input images as legitimate plant specimens with **98% accuracy**. Valid images then proceed to an **EfficientNet-B3 classifier** that identifies the specific crop type (**tomato, potato, or pepper**) with **99% accuracy**. Disease diagnosis is then performed by specialized models: **DenseNet121 architectures** for tomato and potato diseases, and a **ResNet50 model** for pepper diseases ensuring precise pathology identification across different plant families. The final component, a fine-tuned **GPT-2 model**, generates **contextually appropriate treatment recommendations** based on the diagnostic output, achieving strong linguistic performance metrics (**ROUGE-1: 0.82, ROUGE-2: 0.73**).

The solution is deployed as a **smartphone application**. The system offers an analytical platform enabling agricultural companies to monitor disease prevalence patterns and assess treatment efficacy across geographical regions.

This project aims to **mitigate crop losses while promoting sustainable agricultural practices** by delivering accessible AI-based disease detection tools and facilitating enhanced collaboration among farmers, researchers, and agricultural industry stakeholders.

Keywords: *Smart Agriculture, Deep Learning, NLG, Plant Disease Detection, Computer Vision, Mobile Applications, Out-of-Distribution Detection, EfficientNet, DenseNet121, ResNet50, MobileNetV2, GPT-2, Agricultural Decision Support Systems*

ملخص

تتسبب أمراض النباتات في خسائر سنوية تتراوح بين **20% إلى 40%** من الإنتاج الزراعي العالمي، مما يهدد استدامة الزراعة والأمن الغذائي، خاصة في الدول النامية حيث يفتقر المزارعون إلى الخبرة الفيتوباثولوجية والأدوات التشخيصية المتقدمة. تهدف هذه الدراسة إلى تقديم نظام متكامل يعتمد على الذكاء الاصطناعي للكشف عن أمراض النباتات وتقديم توصيات علاجية داعمة، موجهة لكل من المزارعين الأفراد والمؤسسات الزراعية.

يعتمد النظام على هيكل متعدد المراحل، يبدأ باستخدام نموذج **MobileNetV2** للكشف عن الحالات الخارجة عن التوزيع **(Out-of-Distribution)** بدقة تصل إلى **98%**، لضمان أن الصور المدخلة تمثل عينات نباتية حقيقية. تنتقل الصور المقبولة إلى مصنف **EfficientNet-B3** لتحديد نوع المحصول (مثل الطماطم، البطاطس، أو الفلفل) بدقة **99%.** بعد ذلك، تُشخّص الأمراض باستخدام نماذج متخصصة **DenseNet121** :لأمراض الطماطم والبطاطس، و **ResNet50**لأمراض الفلفل، مما يضمن دقة عالية في تحديد الأمراض حسب نوع النبات.

يتولى نموذج لغوي من نوع **GPT-2**، تم تدريبه خصيصًا لهذا الغرض، توليد توصيات علاجية ملائمة بناءً على التشخيص، محققًا أداء لغويًا قويًا وفق مؤشرات**(ROUGE-1 = 0.82 ,ROUGE-2 = 0.73).**

تم تطوير النظام كتطبيق على الهواتف الذكية، يوفر منصة تحليلية تساعد الشركات الزراعية على مراقبة انتشار الأمراض وتقييم فعالية العلاجات في مناطق جغرافية متعددة.

تسعى هذه المساهمة إلى **تقليل خسائر المحاصيل وتعزيز الممارسات الزراعية المستدامة** من خلال تقديم أدوات تشخيصية قائمة على الذكاء الاصطناعي يمكن الوصول إليها بسهولة، وتعزيز التعاون بين المزارعين والباحثين والجهات الفاعلة في القطاع الزراعي.

الكلمات المفتاحية :الزراعة الذكية، التعلم العميق، توليد اللغة الطبيعية، كشف أمراض النبات، الرؤية الحاسوبية، تطبيقات الهاتف المحمول، الكشف عن البيانات الخارجة عن التوزيع، EfficientNet، DenseNet121، ResNet، MobileNetV2، GPT-2، أنظمة دعم القرار الزراعي.

List of Tables

List of Figures

Contents

Chapter 1

General Introduction

1.1 Context and Problem Statement

Agriculture today faces unprecedented challenges due to climate change, depleting resources, and the increasing prevalence of plant diseases. These diseases contribute to crop losses ranging between 20-40% globally each year, posing significant threats to food security and economic stability [1]. In Algeria's Biskra region, where agriculture is a vital economic pillar, annual productions reach approximately **3.47 million quintals of tomatoes**, **1.33 million quintals of peppers**, and **1.2 million quintals of potatoes** [2]. Such disease outbreaks directly affect local food supply and farmer incomes.

Despite the promising advances in artificial intelligence for plant disease detection, several obstacles hinder the practical deployment of these technologies:

- **Technical Limitations:** Variability in field conditions makes it difficult to transfer high-accuracy laboratory models to real-world environments [3]
- **Practical Considerations:** Limited digital literacy among agricultural practitioners and insufficient knowledge support create significant barriers [4, 5]
- **User Diversity:** Current systems often do not accommodate the varied needs of both smallholder farmers and large-scale agricultural enterprises.
- **Pesticide Misuse:** The improper application of pesticides due to inaccurate disease identification leads to environmental degradation, increased resistance in pathogens, and potential health hazards for consumers and agricultural workers [1]
- **Delayed Detection:** Late identification of plant diseases significantly reduces treatment efficacy and increases crop losses, highlighting the critical need for early detection systems that can identify symptoms before widespread damage occurs [5]

These challenges collectively threaten food security and human health on multiple fronts: reduced crop yields impact food availability and economic stability, while pesticide misuse can introduce toxins into the food supply chain and surrounding ecosystems. Early intervention through accurate disease detection is essential for protecting both agricultural productivity and public health

1.2 Motivation and Objectives

The need for accessible AI solutions in agriculture is urgent, given the annual global economic loss of over $220 billion attributed to plant diseases [1]. This challenge is further amplified by the necessity to boost agricultural productivity in anticipation of a global population nearing 10 billion by 2050 [6]. Advances in mobile technology now enable the widespread adoption of agricultural expertise, potentially reducing agrochemical usage by up to 40% while maintaining crop protection efficacy.

The primary objectives are:

1. To develop robust computer vision systems that maintain high performance in variable field conditions.
2. To design intuitive user interfaces accessible to agricultural stakeholders with diverse technical backgrounds.
3. To integrate actionable treatment recommendations seamlessly into the disease detection process.
4. To reduce pesticide misuse through accurate disease identification and appropriate treatment guidance.

To achieve these objectives, we have developed a lightweight, end-to-end mobile application for plant disease detection and recommendation, leveraging advanced deep learning models tailored for both efficiency and accuracy. Specifically, we employ MobileNetV2 for out-of-distribution (OOD) detection and leaf/non-leaf image filtering, EfficientNetB3 for crop classification, DenseNet121 for identifying diseases in potato and tomato crops, and ResNet50 for disease detection in pepper plants. For treatment recommendations and pest management advice, we integrate a GPT-2 based natural language generation module. This model ensemble ensures that the system remains computationally efficient while providing high-quality disease diagnosis and context-aware recommendations.

The system architecture leverages three complementary technological domains:

- Advanced computer vision algorithms for detailed plant image analysis and disease feature extraction.
- Natural language processing capabilities for delivering multilingual, accessible recommendations.
- Optimized deployment strategies that ensure functionality across various environments.

This integrated approach facilitates early disease detection, ensures timely access to appropriate interventions, and supports sustainable agricultural practices. Continuous improvement is enabled through the iterative incorporation of field-generated data, enhancing accuracy and contextual relevance over time.

1.3 Outline

This dissertation is organized into the following chapters:

- **Chapter 2: Smart Agriculture and AI Applications** Reviews the evolution from traditional to smart agriculture and explores AI applications in agriculture sector.
- **Chapter 3: Design and Methodology** Details the proposed system architecture and crop disease detection and recommendation process.
- **Chapter 4: Implementation and Results** Presents system implementation and evaluation results.
- **Chapter 5: Conclusion and Perspectives** Summarizes findings and outlines future directions.

Through these chapters, this work demonstrates how artificial intelligence can be effectively adapted to real agricultural environments, offering robust and practical solutions for plant disease management that promote responsible pesticide use and support early detection to enhance food security and protect human health.

Chapter 2

Smart Agriculture and AI Applications

2.1 Introduction

Agriculture has underpinned human civilization, evolving from manual labor to today's global, industrial-scale systems. However, traditional methods now face mounting challenges such as soil degradation, climate variability, pests, and increased food demand due to population growth [7, 8].

Modern agriculture must deliver higher yields with greater efficiency and less environmental impact. Smart farming technologies address these needs by improving yield prediction, resource management, and climate resilience [3, 9]. The adoption of digital tools has shifted farming from reactive to predictive and prescriptive approaches, guiding better decision-making [10, 11]. As shown in Figure 2.1, today's farms increasingly integrate technology and sustainable practices to boost productivity and reduce ecological footprint.

Figure 2.1: Modern farm landscape [12].

Central to this transformation is the integration of artificial intelligence (AI) technologies, which are driving the shift toward data-driven and intelligent farming systems [13, 14]. AI-powered management increases resilience, optimizes resources, and supports better decision-making crucial as agriculture faces pests, shrinking farmland, and knowledge gaps [3, 15, 7].

2.2 History of Farming

Agriculture has evolved through several major technological eras:

1. **Early Farming (10,000–15,000 BCE):** The Neolithic Revolution transitioned humanity from hunting to organized farming, with the first crop cultivation and animal domestication enabling permanent settlements [16] [17].
2. **Classical and Medieval Farming:** Advances such as metal tools, irrigation, and crop rotation boosted productivity. Agricultural knowledge was systematically recorded in works like Cato's *De Agricultura* and Ibn al-Awwam's *Book of Agriculture* [16] [17].
3. **Industrial Revolution:** Mechanization, including the seed drill, steel plow, and mechanical reaper, transformed farming by increasing yields and reducing manual labor [17] [10].
4. **Green Revolution:** The mid-20th century saw higher yields from new crop varieties and synthetic fertilizers, notably through Norman Borlaug's wheat research. However, these gains sometimes came with environmental costs [8] [18].
5. **Digital Farming Era:** Today, agriculture is shaped by digital technologies precision farming, IoT, remote sensing, big data, and AI enabling data-driven decisions and automation [3] [19].

2.3 Traditional Agriculture

Traditional agriculture represents farming practices that have been developed and refined over thousands of years, relying primarily on manual labor, natural processes, and accumulated knowledge passed down through generations. These methods formed the backbone of agricultural production for most of human history and continue to be practiced by millions of farmers worldwide, particularly in developing regions [20].

2.3.1 Characteristics of Traditional Farming

Traditional agricultural systems are characterized by several fundamental features that distinguish them from modern industrial farming approaches [20]:

- **Manual Labor Dependency:** Traditional farming relies heavily on human and animal labor for most agricultural operations, from land preparation and planting to harvesting and post-harvest processing.

- **Experience-Based Decision Making:** Farmers make decisions based on accumulated knowledge, seasonal patterns, visual observations, and intuitive understanding of local conditions rather than data-driven analysis.
- **Low External Input Use:** Traditional systems typically use minimal external inputs such as synthetic fertilizers, pesticides, or hybrid seeds, instead relying on organic matter, crop rotation, and natural pest control methods.
- **Subsistence-Oriented Production:** Many traditional farming systems focus primarily on producing food for family consumption and local markets rather than commercial export.
- **Biodiversity Conservation:** Traditional farmers often maintain diverse crop varieties and farming systems that support local biodiversity and ecosystem services.

2.3.2 Traditional Farming Practices

Traditional agriculture encompasses a wide range of time-tested practices that have evolved to suit local environmental conditions and cultural preferences [20]:

- **Crop Rotation and Intercropping:** Farmers rotate different crops seasonally and often grow multiple crops together to maintain soil fertility, reduce pest pressure, and maximize land use efficiency.
- **Natural Fertilization:** Use of animal manure, compost, green manure, and other organic materials to maintain soil fertility without synthetic chemicals.
- **Indigenous Seed Varieties:** Cultivation of locally adapted, open-pollinated seed varieties that have been selected and preserved by farming communities over generations.
- **Integrated Pest Management:** Traditional methods include companion planting, beneficial insect habitat, botanical pesticides, and cultural practices to control pests and diseases.
- **Water Conservation Techniques:** Traditional irrigation methods such as terracing, contour farming, and rainwater harvesting that work in harmony with natural water cycles.
- **Seasonal Timing:** Planting, cultivation, and harvesting activities are timed according to traditional calendars, lunar cycles, and local weather patterns.

2.3.3 Limitations of Traditional Agriculture

While traditional agriculture has sustained human populations for millennia, it faces significant challenges in meeting modern agricultural demands [20] [21]:

- **Lower Productivity:** Traditional methods generally produce lower yields per unit area compared to modern intensive farming systems.
- **Labor Intensity:** Heavy reliance on manual labor makes traditional farming physically demanding and time-consuming, limiting scalability.
- **Weather Vulnerability:** Traditional systems are often more vulnerable to extreme weather events and climate variability due to limited adaptive technologies.
- **Limited Market Access:** Traditional farmers may face challenges in accessing modern markets, credit systems, and value-added processing opportunities.
- **Knowledge Transfer Challenges:** Traditional knowledge may be lost as younger generations migrate to urban areas or adopt modern practices.
- **Pest and Disease Management:** Limited options for controlling serious pest outbreaks or disease epidemics that can devastate entire harvests [22].
- **Inconsistent Quality:** Lack of standardization in traditional practices can lead to variable product quality and market acceptance issues.

2.3.4 Traditional Agriculture in the Modern Context

As global agriculture faces sustainability challenges, there is growing recognition of the value of traditional knowledge and practices. Many researchers and policymakers advocate for integrating the best aspects of traditional agriculture with modern technologies to create more sustainable and resilient farming systems [23] [20].

Figure 2.2: Traditional farming landscape showing manual labor and simple tools [24].

This integration approach, often called "agroecology" or "sustainable intensification," seeks to combine traditional ecological principles with modern scientific understanding to develop farming systems that are both productive and environmentally sound [25]. Understanding traditional agriculture provides crucial context for appreciating the revolutionary changes brought about by smart agriculture technologies, which we will explore in the following section.

The transition from traditional to smart agriculture represents not just a technological shift, but a fundamental transformation in how farmers interact with their crops, make decisions, and manage agricultural systems. While traditional agriculture will continue to play an important role, especially in developing regions, the integration of smart technologies offers unprecedented opportunities to address the growing challenges of food security, environmental sustainability, and climate adaptation.

2.4 Smart Agriculture

Smart agriculture, also known as Agriculture 4.0, integrates advanced technologies to optimize and transform farming practices [14] [11] by enabling precise monitoring and management of crops, soil, and resources. Its primary objectives are to enhance productivity, promote sustainability, and increase the resilience of agricultural systems.

2.4.1 Traditional vs. Smart Farming

While traditional agriculture relies on manual labor and experience-based decision making, smart agriculture employs modern information and communication technologies (ICT) such as cloud computing, big data, robotics, automation, and IoT to enable continuous data collection, real-time monitoring, automated device communication, and data-driven decisions [4]. This paradigm shift allows for more efficient resource management and proactive responses to field challenges, moving from manual intervention to automation, predictive analytics, and precision management [10] [18].

Traditional Agriculture

- Relies on manual labor and direct observation of crops and soil.
- Makes decisions based on farmers' experience and intuition.
- Applies water and fertilizers evenly across all fields.
- Responds to issues only after problems become visible.
- Collects minimal data, often limited to simple field notes.

Smart Agriculture

- Uses automated sensors to continuously monitor crops and soil.
- Bases decisions on real-time data and predictive algorithms.
- Delivers water and fertilizers precisely where and when needed.
- Detects and addresses issues proactively using early warning systems.
- Integrates and analyzes comprehensive data from multiple sources.

Figure 2.3: Traditional farming practices[12].

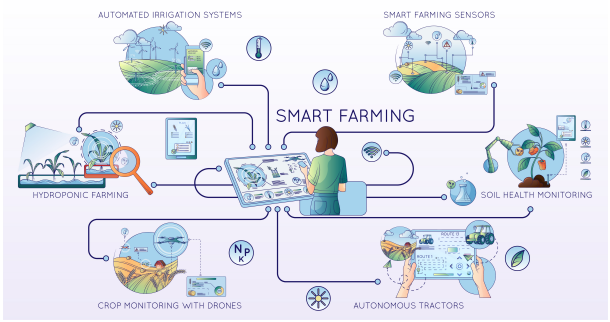


Figure 2.4: Modern smart farming technologies[26].

Figure 2.5: Comparison between Traditional and Smart Agriculture.

2.4.2 Technologies in Smart Farming

Smart agriculture systems incorporate a range of technologies to improve productivity, resource efficiency, and sustainability [9]:

- **Internet of Things (IoT):** Connects devices such as soil sensors and automated machinery for real-time monitoring of crop and environmental conditions [19] [13].
- **Artificial Intelligence (AI):** Applies machine learning and computer vision to analyze data from sensors and drones, supporting yield prediction, and optimization [27] [28] [29].

- **Drones and Remote Sensing:** Provides high-resolution imagery for crop health monitoring and early detection of diseases or nutrient deficiencies [30].
- **Farm Automation and Robotics:** Includes autonomous tractors, robotic harvesters, and precision sprayers, reducing manual labor and increasing efficiency [4] [31].
- **Communication Networks and Software Solutions:** Cloud platforms and farm management software integrate and analyze data from various sources, facilitating real-time decision making [13].
- **Digital Twins:** Virtual models of farms support simulation and optimization of agricultural operations [11] [14].
- **Big Data Analytics:** Extracts insights from large datasets to inform decisions and reveal important patterns [3].
- **Fusion Data Augmentation:** Combines multiple data sources to enhance machine learning model accuracy, especially for plant disease detection [32].

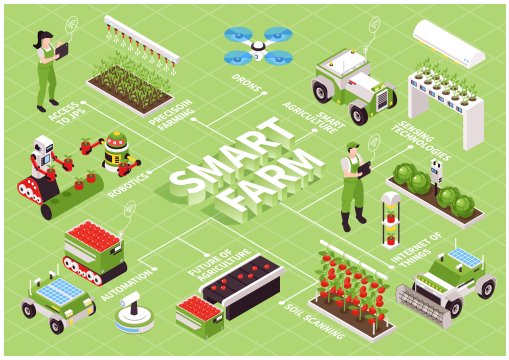


Figure 2.6: Smart farming technologies [26].

2.4.3 Advantages of Smart Agriculture

Smart agriculture provides several benefits over traditional methods:

- **Optimization of Resource and Input Consumption:** Automated systems use water, energy, and fertilizers efficiently, minimizing waste and environmental impact [14] [10].
- **Higher Yields at Lower Costs:** Precision practices and data-driven decisions increase productivity and reduce expenses [3] [18].
- **Better Land Utilization:** Technology enables efficient use of available space, including areas otherwise underutilized [9].

- **Reduced Manual Labor:** Robotics and automation decrease the need for physical labor, increasing operational efficiency [4] [31].
- **Real-Time Monitoring and Predictive Analytics:** Continuous data tracking and analytics anticipate issues and optimize management [19] [13].
- **Improved Decision Making:** Big data analytics and AI offer actionable insights for planning and operations [27] [28].
- **Sustainability and Environmental Protection:** Smart systems reduce resource consumption and environmental impact, supporting long-term sustainability [11] [25].

2.4.4 Smart Agriculture Challenges

Despite technological progress, agriculture faces ongoing challenges:

- **Climate Change:** Changing rainfall and more frequent extreme weather reduce crop yields [15].
- **Water and Land Issues:** Water scarcity affects 40% of cropland, and arable land declines due to urbanization and soil degradation [33].
- **Crop Diseases:** Plant diseases cause global losses exceeding $220 billion annually [34] [22].
- **Population Growth:** The global population is projected to reach 9.7 billion by 2050, increasing food demand and straining agricultural systems [7] [6].

Figure 2.7: Smart farming challenges [35].

2.5 Crop Disease Management

The following examples focus on three economically important crops vulnerable to disease, where early detection systems can be particularly beneficial.

- **Potato**

Potatoes, one of the world's most consumed food crops, are highly susceptible to diseases that can devastate entire fields rapidly, making early detection essential [36] [37]. Computer vision systems have demonstrated efficacy in detecting these diseases early [38] [39].

Figure 2.8: Viral infection in potato [37].

- **Tomato**

Tomatoes are prone to fungal, bacterial, and viral diseases [40] [41]. Automated monitoring using computer vision enables early detection of symptoms, playing a crucial role in preserving yield and product quality [39].

Figure 2.9: Late blight in tomato [41].

- **Pepper**

Pepper crops face viral diseases like the mosaic virus, which can spread rapidly [42]. Early identification of characteristic symptoms such as leaf mottling is essential for timely intervention.

Figure 2.10: Mosaic virus in pepper [42].

2.6 AI Techniques in Agricultural Applications

As agriculture embraces digital transformation, AI has emerged as a cornerstone technology for addressing crop disease challenges and enhancing productivity. AI technologies enable precise monitoring, predictive analysis, and automation of complex tasks traditionally dependent on manual labor and experiential knowledge.

2.6.1 Machine Learning

Machine learning, a subset of artificial intelligence that enables systems to learn from data and improve over time without explicit programming, has substantially advanced data-driven decision-making in modern agriculture [27] [43] [44].

- **Supervised Learning**
  Supervised learning trains models using labeled datasets to perform classification or regression tasks. These algorithms demonstrate robust performance in agricultural applications when labeled data is available [45] [46].
  - **Random Forests (RF)**: An ensemble learning method used for crop prediction and disease detection.
  - **Support Vector Machines (SVM)**: Effective for plant disease classification and image analysis, particularly with small datasets [47] [48].
  - **XGBoost**: A gradient boosting framework that has shown success in improving irrigation efficiency [28].
- **Unsupervised Learning**
  Unsupervised learning identifies patterns in unlabeled datasets, useful for exploratory data analysis when labeled data is scarce [49] [30].
  - **K-Means Clustering**: Used in field zoning and soil classification tasks [50] [51].
  - **Anomaly Detection**: Facilitates early identification of irregularities in crop growth or disease symptoms.
- **Semi-supervised Learning**
  Combines supervised and unsupervised methods using both labeled and unlabeled data. Applied in remote sensing and plant pathology with reduced manual annotation requirements [52] [53].
- **Reinforcement Learning**
  Uses trial-and-error learning where agents maximize cumulative rewards through environmental interactions. Applied in autonomous greenhouse control, irrigation scheduling, and precision spraying [54] [55].
- **Transfer Learning**
  Reuses knowledge from pre-trained models for new tasks, particularly valuable when domain-specific data is limited. Achieves high accuracy with minimal training data for disease detection and yield estimation [56] [57].

2.6.2 Deep Learning

Deep learning processes information through hierarchical layers of artificial neural networks, enabling models to automatically learn complex patterns from large datasets. Recent advances in computing power have positioned deep learning at the forefront of agricultural AI [58] [59] [60].

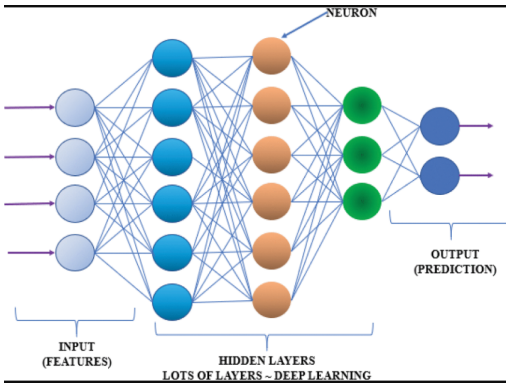


Figure 2.11: Deep neural network architecture [47].

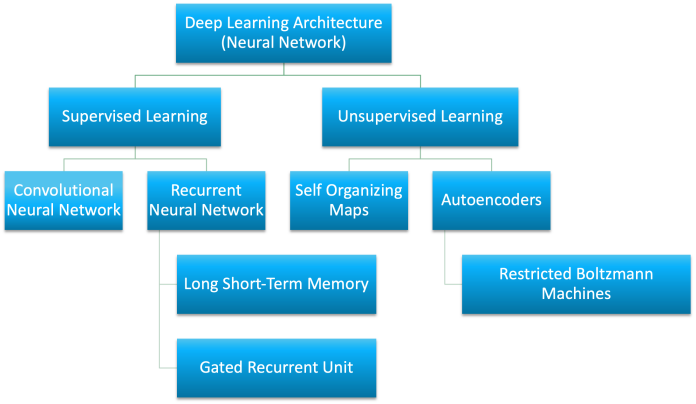


Figure 2.12: Deep learning architecture branches [61].

- **Supervised Deep Learning**
  Deep supervised learning models are trained on labeled datasets with the capacity to automatically extract hierarchical features from raw data.
  - **Convolutional Neural Networks (CNNs)**: Excel at image pattern recognition for plant disease identification.

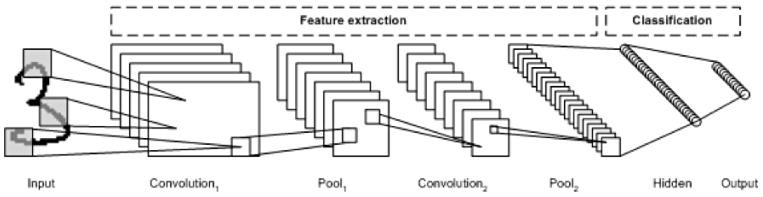


Figure 2.13: CNN architecture for image analysis [61].

  - **Recurrent Neural Networks (RNNs)**: Process sequential data for time-series analysis [62].
    - **LSTM (Long Short-Term Memory)**: Effective for long-term pattern recognition in crop growth prediction [63].
    - **GRU (Gated Recurrent Units)**: Faster alternative for irrigation scheduling tasks [64].

- **Unsupervised Deep Learning**
  Discovers hidden patterns in unlabeled data through neural networks that capture complex, non-linear relationships.
  - **Self-Organizing Maps (SOMs)**: Create visual maps grouping similar plant or soil data [65].
  - **Autoencoders**: Learn data representations by reconstruction, including Restricted Boltzmann Machines for anomaly detection [66].

- **Deep Semi-supervised Learning**
  Applies neural networks to leverage both labeled and unlabeled data, particularly valuable for agricultural remote sensing [52].

- **Deep Reinforcement Learning**
  Combines neural networks with reinforcement learning for sophisticated decision-making in autonomous robots and greenhouse management [54].

- **Deep Transfer Learning**
  Uses CNNs pretrained on large datasets like ImageNet, achieving high performance in plant disease identification with less labeled data [56].

2.6.3 Classification vs. Clustering

Classification is a supervised learning process where models predict predefined categories using labeled datasets. CNNs have demonstrated high efficacy in plant disease diagnosis [67].

Classification types include:

- **Binary Classification**: Predicting one of two outcomes (healthy vs. diseased plants).
- **Multi-class Classification**: Assigning inputs to one of multiple exclusive classes (specific disease types).
- **Multi-label Classification**: Predicting multiple labels for single inputs (multiple stress factors).

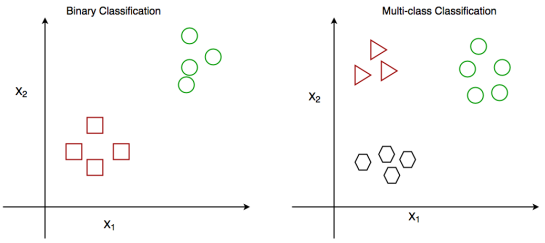


Figure 2.14: Binary vs. multi classification types [68].

Clustering is an unsupervised learning approach grouping data points based on similarities without prior class labels [30] [49].

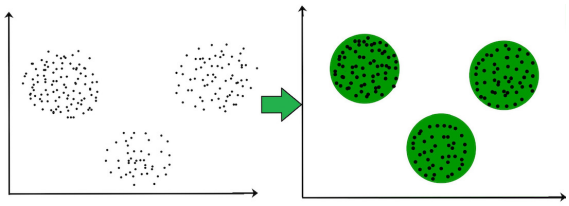


Figure 2.15: Clustering vs. classification approaches [68].

This project uses a multi-model classification pipeline including MobileNetV2-based out-of-distribution detection, EfficientNet-B3 crop classification, and specialized disease classification models.

2.6.4 Natural Language Processing (NLP)

Methods are increasingly integrated into agricultural AI systems, offering powerful tools for text-based information extraction, decision-making support, and interactive advisory generation [69] [70]. NLP plays a critical role in tasks such as:

- **Text Classification**: Categorizing agricultural documents based on topics such as soil health, pest threats, or advisory urgency [71].
- **Named Entity Recognition (NER)**: Automatically identifying and labeling key entities like plant varieties, pathogens, chemicals, and geographical locations in scientific texts [72].
- **Information Retrieval**: Extracting structured knowledge from unstructured data sources such as journals, manuals, or farm logs to assist agronomists and AI systems alike [70].

As depicted in Figure 2.16, NLP frameworks integrate both Natural Language Understanding (NLU) and Natural Language Generation (NLG) components to create comprehensive systems for processing and generating human language. This architecture is particularly valuable for agricultural advisory systems.

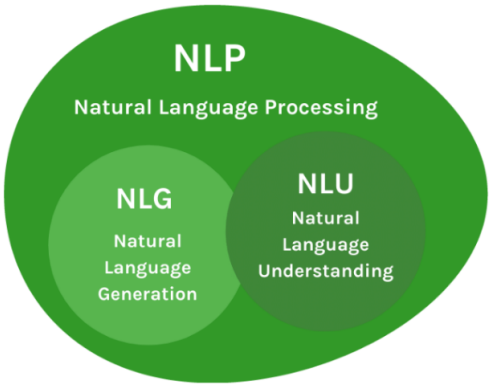


Figure 2.16: NLP framework integrating NLU and NLG [73].

- **Natural Language Understanding (NLU)**
  Forms the foundational layer of NLP systems, enabling machines to comprehend the semantics and context of human language. In agricultural AI, NLU helps bridge the gap between raw text and actionable insights by enabling systems to:
  - **Intent Detection**: Understanding the underlying goals behind farmers' queries, such as seeking pest control advice or weather forecasts.
  - **Entity Linking**: Associating detected entities (like crops, diseases, or fertilizers) with relevant knowledge bases to enrich interpretation.

  - **Semantic Parsing**: Translating complex natural language inputs into structured formats interpretable by downstream decision-support models.

  NLU ensures that agricultural advisory systems correctly interpret the meaning behind user inputs, enhancing the relevance, personalization, and effectiveness of generated recommendations [74] [68].

- **Natural Language Generation (NLG)**
  Focuses on automatically producing coherent textual content from structured data or model outputs. In agriculture, NLG facilitates the translation of complex analytical results into actionable, farmer-friendly language [75] [76].
  Key applications include:

  - **Automated Report Generation**: Producing pest risk summaries, irrigation schedules, or yield forecasts based on sensor data and model predictions.
  - **Personalized Advisory Generation**: Tailoring recommendations to the specific needs, crop types, or environmental conditions of individual users.
  - **Multilingual Accessibility**: Providing farmers with localized content in their native language, ensuring better adoption and understanding.

  In this project, NLG serves an important role in **generating structured recommendation reports** based on model outcomes especially when disease predictions or environmental readings need to be transformed into practical advice for end users [75] [77] [76].

2.7 Related Works

Recent advancements in artificial intelligence (AI) for agriculture have transitioned from academic explorations to impactful real-world applications across a spectrum of technical domains. Computer vision and deep learning techniques, for instance, have achieved remarkable accuracy in plant disease identification. For example, Mohanty et al. (2016)[78] reported 99.35% accuracy under controlled conditions, but only 31% in real-world field tests. This gap highlights the persistent challenge of model generalization beyond lab environments.

A significant advancement in plant disease detection was presented at IEEE_ICT (2024)[29], where researchers developed a tomato disease classification system achieving 95.31% accuracy while incorporating contextual treatment recommendations for identified pathogens. Complementing this work, Kant et al. (2024)[79] demonstrated that models based on EfficientNetB3 architecture provide an optimal balance between computational efficiency and diagnostic accuracy (92.25%) through a multilabel classification approach. This balance is particularly valu-

able for real-world agricultural applications where resource constraints exist, making sophisticated disease detection technology increasingly accessible and practical for farmers operating in diverse environments.

To overcome the constraints of single-model systems, recent research has explored hybrid architectures. For instance, Shaheed et al. (2023)[80] combined ResNet50 with Vision Transformer components, resulting in over 97.65% accuracy for potato disease classification and enhanced robustness to image quality variations. Similarly, Wang et al. (2024)[81] proposed a specialized architecture tailored for greenhouse environments, reporting 92.3% precision in complex backgrounds. However, this approach introduced additional implementation complexity.

Another direction leverages the synergy between IoT sensor data and AI. Sravanthi et al. (2024)[5] combined real-time IoT data with historical analytics and Natural Language Processing (NLP), achieving 99% accuracy in crop recommendations and a 97.8% comprehension rate among farmers. This demonstrates the critical role of both data integration and effective communication in agricultural AI systems.

Transfer learning is widely adopted to address the scarcity of labeled agricultural datasets. Aversano et al. (2020)[82] utilized VGG-based transfer learning for tomato disease classification, attaining 94.7% accuracy and reducing training time. Likewise, Moid et al. (2021)[56] applied the Xception architecture within a transfer learning framework, improving generalization across diverse environmental conditions.

Study	Primary Focus	Performance	Key Strengths	Limitations
[78]	Image-based plant disease detection	99.35% controlled, 31% real-world	High accuracy under controlled conditions	Poor generalization to real-world data
[5]	IoT-driven recommendation	99% accuracy	Real-time decision support system	Depends on optimal sensor placement
[29]	Tomato disease detection on mobile	95.31% accuracy	User-friendly mobile interface, detailed reports	Execution time bottlenecks
[83]	Multi-dataset DL approach	80.19%	Dataset diversity, improved robustness	Lacks detailed performance metrics
[80]	Multi-architecture detection	97.65% accuracy	Robust to image quality variations	High computational demands
[82]	Tomato disease VGG-based transfer learning	97.3% accuracy	Reduced training time	Limited generalization to field conditions
[84]	Multi-level detection for potato	99.75% accuracy	Early vs. late blight distinction	Region-specific training data
[85]	Tomato leaf disease detection	91.2% accuracy	Works in varied lighting	Optimized for greenhouse only
[38]	CNN-based potato leaf disease detection using sequential model	94.2% accuracy	Effective sequential modeling for improved detection	Evaluation limited to conference dataset, needs real-world validation
[81]	Greenhouse detection	92.3% mAP	Complex background handling	Complex implementation requirements
[79]	Multilabel plant disease classification	92.25% accuracy	Capable of detecting multiple diseases per image	Potential challenges in resource-constrained agricultural settings

Table 2.1: Comparative analysis of AI-Based plant disease detection studies.

2.8 Synthesis

Modern agriculture is limited by three core challenges that constrain the deployment of AI systems in real-world field settings. Environmental variability in farms such as inconsistent lighting, varied backgrounds, and unpredictable disease symptoms reduces the effectiveness of AI models trained in controlled laboratory conditions. As shown by Mohanty et al. [78], classification accuracy can drop from 99.35% in the lab to just 31% in field conditions, revealing a critical gap between research and application. In parallel, widespread digital divides across agricultural communities hinder access to AI systems. Many farmers lack reliable infrastructure and experience with digital tools, making it difficult to adopt complex solutions like those in Shaheed et al. [80], which achieved 97.65% accuracy but are too computationally demanding for most mobile devices. Moreover, many existing systems stop at disease identification and fail to provide actionable treatment guidance, limiting their real-world usefulness. Even when systems like those presented at IEEE_ICT [29] achieve high accuracy, their performance can vary in different usage environments.

This study proposes a multi-model AI system architecture that addresses these limitations through careful design and a focus on accessibility, reliability, and usability. The system begins with an out-of-distribution detection module based on MobileNetV2 to verify that input images are relevant plant specimens before disease detection is attempted. This component improves reliability and prevents common misclassifications caused by irrelevant or low-quality inputs.

The next component is a multi-crop classification stage using EfficientNet-B3. This model distinguishes between tomato, potato, and pepper plants to route the image to the appropriate disease detection model. This approach builds upon recent advances in multilabel classification techniques demonstrated by Kant et al. [79], who achieved 92.25% accuracy with EfficientNetB3 for detecting multiple plant diseases simultaneously. Their work highlights the importance of computational efficiency in real-world agricultural applications, a principle that guides our multi-model architecture design. Unlike generic multi-dataset classifiers like the one in Krishna et al. [83], which achieved only 80.19% accuracy, this step ensures that diagnosis is tailored to the specific crop. For disease detection, the system uses DenseNet121 for tomato and potato and ResNet50 for pepper diseases. This crop-specific specialization acknowledges the unique disease patterns of each plant family and improves upon generalized approaches such as that in Agarwal et al. [85], which reached only 91.2% in greenhouse settings.

The final stage of the system is a bilingual recommendation engine built using a fine-tuned GPT-2 model. This component translates disease classification results into clear, actionable treatment suggestions in multiple languages. It addresses the gap left by most reviewed systems, which fail to provide usable instructions for farmers, leading to improper pesticide use

and poor intervention decisions.

This proposed system creates an end-to-end workflow from image capture to treatment advice, aligning AI performance with real-world agricultural needs. While systems like Bonik et al. [38] achieved 94.2% accuracy without field validation, this study emphasizes practical deployment from the outset. The approach prioritizes real-world reliability and supports the needs of smallholder farmers, who represent the majority of the global agricultural workforce.

2.9 Conclusion

The transition to smart agriculture transcends the adoption of new technologies it fundamentally reimagines farming practices for the 21st century by merging traditional agricultural knowledge with modern digital approaches to enhance productivity while promoting sustainability. Early disease detection using artificial intelligence stands out as one of the most promising applications, where deep learning and computer vision techniques enable identification of plant diseases at initial stages, thereby reducing crop losses and minimizing pesticide use.

As demonstrated through the visual disease symptoms analysis, AI technologies show significant potential in agricultural applications. However, substantial performance gaps persist between controlled laboratory conditions and real-world field deployment, with additional limitations in computational efficiency, input validation, and actionable guidance provision. While individual AI components achieve promising results, comprehensive end-to-end systems addressing both technical robustness and practical usability remain insufficient.

This necessitates the development of integrated AI architectures that bridge laboratory performance with field deployment requirements while providing accessible solutions for diverse farming contexts. The next chapter details the proposed architecture and its practical, scalable design to address these identified limitations in agricultural AI systems.

Chapter 3

Design and Methodology

3.1 Introduction

Following the analysis of existing agricultural challenges and AI-based solutions, we propose a methodology focuses on developing a modular and scalable plant disease detection and recommendation system tailored for real-world agricultural use. The design integrates image validation, crop and disease classification, and bilingual recommendation generation, with an emphasis on mobile compatibility, practical usability, and adaptability to diverse crops, diseases, and user needs.

3.2 Proposed Approach

Traditional agriculture faces mounting pressure from increasing global population demands, with food security becoming a critical concern as populations continue to grow. Conventional farming practices, characterized by manual labor, experience-based decision making, and reactive approaches to crop diseases, result in significant global annual losses due to plant diseases alone. These losses translate to substantial economic damage, threatening both food security and farmer livelihoods.

The limitations of traditional agricultural practices are particularly evident in disease management, where farmers often rely on visual inspection and empirical knowledge to identify plant diseases. Late detection, misdiagnosis, and inappropriate treatment applications frequently occur, resulting in reduced crop yields and increased pesticide usage.

The integration of artificial intelligence into agricultural systems presents unprecedented opportunities to address these challenges through precision farming, real-time monitoring, and data-driven decision support. Current AI-based agricultural solutions face significant deploy-

ment barriers including poor generalization from laboratory to field conditions, limited accessibility for farmers with varying technical literacy levels, and inadequate integration of disease detection with actionable treatment recommendations.

We aim to develop an integrated solution that brings expert-level plant disease diagnosis directly to the field through accessible mobile technology. Our approach addresses the critical need for early disease detection while ensuring practical applicability across diverse agricultural contexts.

As illustrated in Figure 3.1, our proposed system implements a comprehensive architecture designed to transform plant leaf images into actionable disease diagnoses and treatment recommendations.

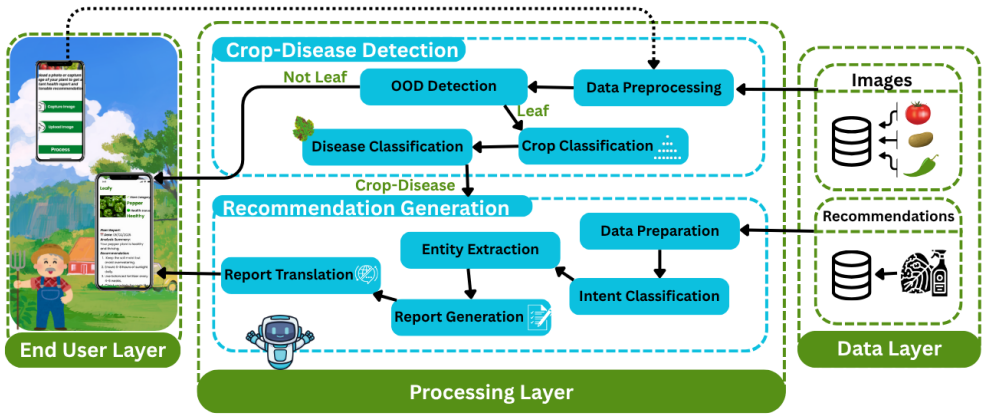


Figure 3.1: Proposed approach for crop disease detection and recommendation generation system.

The proposed system, as shown in Figure 3.1, implements a system architecture designed to transform plant leaf images into actionable disease diagnoses and treatment recommendations:

- **End User Layer:** Serves as the primary interface between farmers and the AI system through an intuitive mobile application designed for agricultural workers with varying levels of technical literacy. The mobile interface provides:
  - Image capture capabilities through the device camera.
  - Photo gallery integration for selecting existing plant images.
  - Clear visualization of diagnostic results.
  - Farmer-friendly presentation of treatment recommendations in accessible language.
  - User-centered design with simple navigation and visual feedback.

- **Processing Layer:** Functions as the analytical core of the system and contains two primary components that work sequentially to analyze plant images and generate recommendations:
  - *Crop Disease Detection:* Implements a sequential analytical pipeline that processes images through:
    - Data Preprocessing: Standardizing image inputs such as resizing and normalization.
    - Out-of-Distribution (OOD) Detection: Validating whether the image contains valid plant leaf material.
    - Decision Branching: Routing non-leaf images back to the user with appropriate feedback.
    - Crop Classification: Identifying the specific plant species (tomato, potato, or pepper).
    - Disease Classification: Applying crop-specific disease diagnosis models to identify pathologies.
  - *Recommendation Generation:* Transforms diagnostic outputs into practical treatment advice through:
    - Data Preparation: Structuring diagnostic results for recommendation generation.
    - Entity Extraction: Identifying relevant components such as crop type and disease name.
    - Intent Classification: Determining appropriate recommendation strategies based on disease context.
    - Report Generation: Creating structured treatment guidelines with technical details.
    - Report Translation: Converting technical information into accessible, actionable language for farmers.
- **Data Layer:** Provides the knowledge repositories that support both analytical processing and recommendation generation:
  - Images Repository: Contains structured datasets of plant images used for model training and validation, organized by crop type and disease categories. The reposi-

tory includes visual examples of healthy and diseased specimens for each supported crop type.

– Recommendations Knowledge Base: Maintains comprehensive crop-disease-treatment mappings including causal factors, symptoms progression patterns, treatment options with effectiveness ratings, and preventive measures. This knowledge base links specific pathologies to contextually appropriate intervention strategies with supporting agricultural science references.

The proposed system architecture facilitates bidirectional data flow from right to left for developers managing the knowledge repositories and model optimization, and from left to right for end users submitting images and receiving diagnostic results. This design ensures both operational efficiency and a seamless user experience.

3.3 System Functionalities

To better explain the system's behavior and structure, we have developed comprehensive UML diagrams that document both the static structure and dynamic interactions within the system. These diagrams serve as essential artifacts in the development process, providing a visual blueprint for implementation while facilitating clear communication between technical and non-technical stakeholders.

3.3.1 Use Case Analysis

The Use-Case Diagram in Figure 3.2 provides a comprehensive view of our Leafy App's functional architecture by illustrating two key actor types the farmer (User) and the Google Cloud Backend and their interactions with the system.

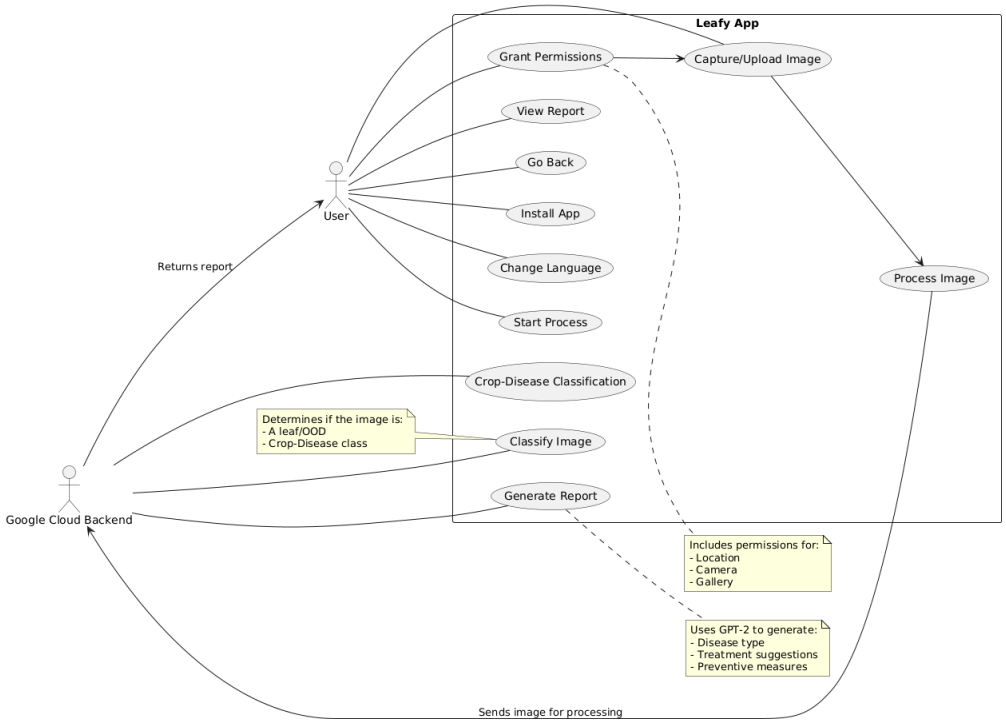


Figure 3.2: System use case diagram.

For farmers, the diagram showcases essential operations involving the End User Layer including:

- Granting permissions (for location, camera, and gallery access).
- Capturing or uploading plant images.
- Viewing diagnostic reports.
- Customizing their experience through language preferences.

The backend services implement the Processing Layer functionality through:

- Image processing and enhancement
- Leaf/Out-of-Distribution (OOD) detection to validate input.
- Crop Disease Classification using specialized models.
- Report generation using GPT-2 to produce a comprehensive output that includes:
  - Identified crop.
  - Detected disease.
  - Likely cause of the disease.

- Observable symptoms.
- Recommended treatment.

As shown in Figure 3.2, the diagram explicitly shows dependencies, such as how image capture must precede processing, and how classification determines if the image contains a valid leaf before identifying the specific crop-disease combination. This representation clearly delineates the distribution of intelligence between the mobile client and cloud backend.

3.3.2 Activity Diagram Analysis

The Activity Diagram in Figure 3.3 details the complete operational workflow of our plant disease detection application from user initiation to diagnostic results.

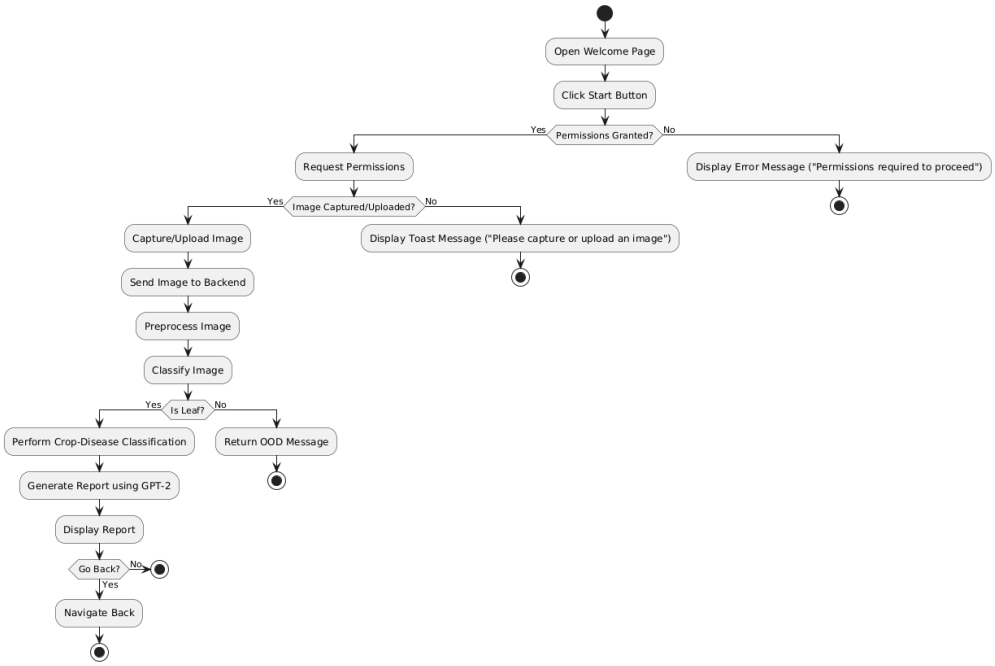


Figure 3.3: System activity diagram.

The diagram identifies three critical decision points that determine the user's path through the system:

- **Permission verification:** Ensures the application has necessary access to device resources.
- **Image confirmation:** Validates that a usable image has been provided.
- **Leaf validation:** Confirms the image contains analyzable plant material through Out-of-Distribution (OOD) detection.

When permissions are denied or images aren't provided, the system displays appropriate error messages and terminates processing, preventing wasted computational resources. As illustrated in Figure 3.3, the sequential pipeline operations mirror the architecture described in Figure 3.1:

1. Image preprocessing for standardization.
2. Classification through the multi-stage Crop Disease Detection process.
3. GPT-2 Recommendation Generation based on knowledge base information.

Upon displaying results, the diagram shows the navigation options available to farmers, enabling them to either analyze another plant sample or exit the application. This comprehensive workflow visualization ensures our implementation delivers a responsive, user-centered experience that addresses potential failure points while guiding users through the complete disease diagnosis and recommendation system process.

3.4 Crop Disease Detection and Recommendation Process

As shown above in(Figure 3.1), our solution comprises two key functional components within the Processing Layer: Crop Disease Detection and Recommendation Generation. These components work in concert to transform raw plant images into actionable agricultural advice.

Figure 3.4 illustrates the processing pipeline of the system.

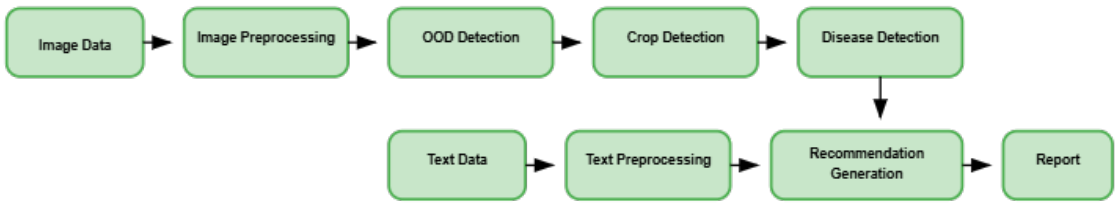


Figure 3.4: Crop disease detection and recommendation generation process.

As shown in Figure 3.4, the workflow begins with raw image data input from the End User Layer, which undergoes preprocessing to standardize visual information. The processed images then enter Out-of-Distribution (OOD) detection to validate plant material presence. Crop classification follows, identifying specific plant species and routing images to specialized disease detection models optimized for each crop type. Text data undergoes parallel preprocessing to

structure the knowledge base for recommendations. The Recommendation Generation component transforms technical diagnoses into farmer-friendly advice, culminating in a comprehensive report combining visual and textual information that is delivered back to the End User Layer.

Figure 3.5 presents a more detailed view of the technical architecture of the system's Processing Layer, expanding on the processes outlined in Figure 3.4.

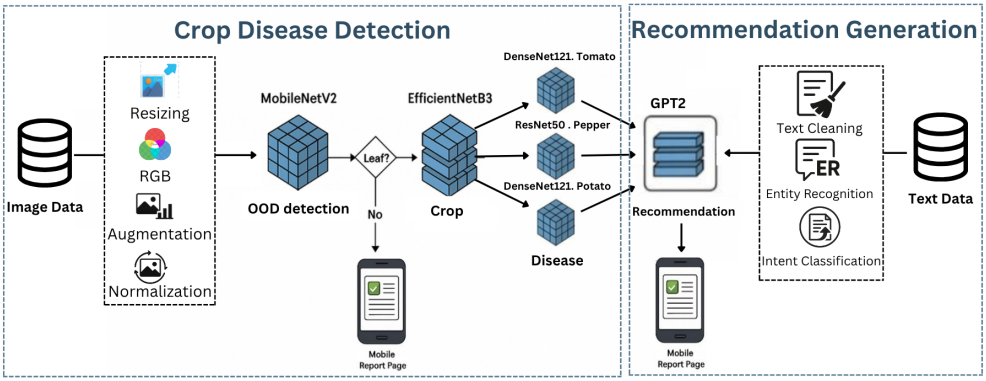


Figure 3.5: Crop disease detection and recommendation generation system detailed process.

As presented in Figure 3.5, the Crop Disease Detection process, starting with image data from the Data Layer, followed by preprocessing steps: resizing, RGB normalization, augmentation, and standardization. Preprocessed images undergo analysis through neural networks, beginning with MobileNetV2 for leaf detection to filter out non-leaf images. For confirmed leaf images, EfficientNetB3 performs crop classification to identify tomato, pepper, or potato plants. Based on classification results, the system routes images to crop-specific disease classification models: DenseNet121 for tomato and potato diseases, and ResNet50 for pepper diseases.

The right part of Figure 3.5 details the Recommendation Generation subsystem built around a fine-tuned GPT-2 language model. Text data passes through cleaning to standardize input, entity recognition to identify key agricultural elements, and intent classification to determine recommendation context. The GPT-2 model generates structured treatment recommendations including disease identification, causes, symptoms, and remediation strategies. The output appears as a mobile-friendly report for farmers to implement in their fields, delivered through the End User Layer.

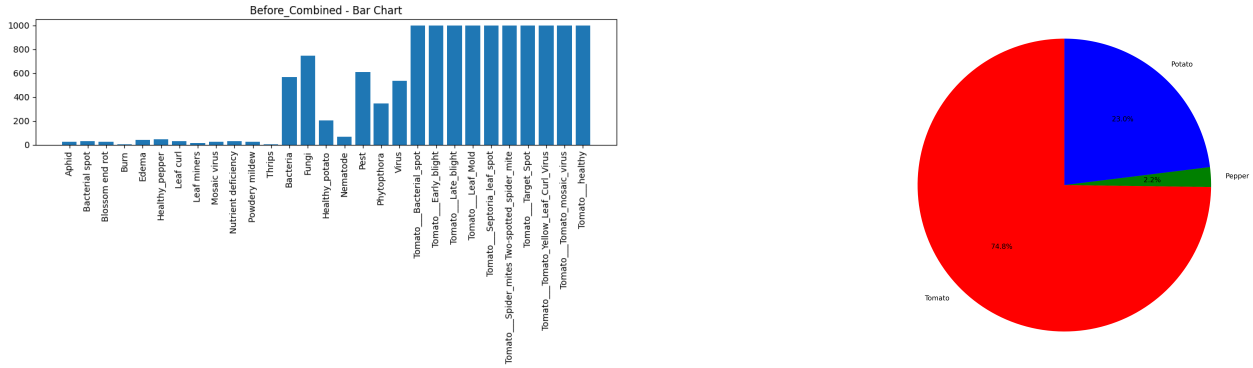
3.4.1 Data Pipeline

Our data pipeline encompasses the collection and processing of both image and text data to support the model training and inference processes within the Processing Layer.

- **Data Collection**
  The training data for the Crop Disease Detection models consists of public datasets and manually collected images stored in the Images Repository of the Data Layer. The collection includes:
  - **Tomato Plants:** 10,000 images across nine disease categories.
  - **Potato Dataset:** 3,080 images covering seven classes.
  - **Pepper Dataset:** 290 images encompassing 12 disease categories.

  An imbalance exists predominantly due to the high volume of tomato samples, this imbalance is addressed using data augmentation and resampling. Two supplementary datasets are included in the Data Layer:
  - **Recommendation Dataset:** Mapping crop-disease-treatment for GPT-2 fine-tuning of the Recommendation Generation component.
  - **Out-of-Distribution (OOD) Detection Dataset:** Contains in-distribution and out-of-distribution samples for training the leaf detection model.

  Figure 3.6 illustrates the dataset distribution before augmentation in the Data Layer's Images Repository, showing both the combined dataset distribution (Figure 3.6a) and the overall dataset size distribution (Figure 3.6b).



(a) Combined dataset distribution.

(b) Dataset size distribution.

Figure 3.6: Dataset distribution before augmentation.

Our comprehensive data preprocessing framework within the Processing Layer addresses two distinct data modalities: images for Crop Disease Detection and text for Recommendation Generation. Each modality follows a specialized pipeline to ensure standardized inputs and enhanced feature quality.

- **Data Preprocessing**
  Data preprocessing constitutes a critical foundation of our system, ensuring optimal model performance through systematic preparation of both visual and textual inputs for the crop disease detection and recommendation pipeline.

  - **Image Data Processing**
    We implemented a robust image preprocessing pipeline within the Crop Disease Detection component with multiple stages:

    * **Resizing:** All images are resized to 224×224 pixels.
    * **RGB Conversion:** All images are converted to the RGB color space, ensuring consistent 3-channel representation regardless of original format.
    * **Normalization:** Pixel values are scaled to the [0,1] range.
    * **Data Augmentation:** Techniques such as random flipping, rotation (up to 20%), zoom, and contrast adjustments are applied.
    * **Class Balancing:** Oversampling is used for underrepresented classes.

    Figure 3.7 illustrates our comprehensive image preprocessing workflow within the Processing Layer. These processes standardize input features while preserving disease-relevant characteristics essential for accurate Crop Disease Detection.

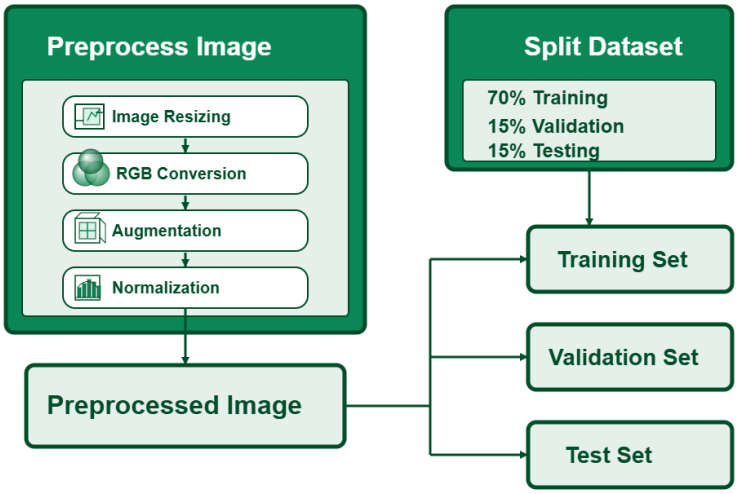


Figure 3.7: Image preprocessing workflow.

– **Text Data Processing for NLG**

For textual data used in the Recommendation Generation component, particularly focusing on plant disease entities, we implemented a multi-stage preprocessing pipeline:

* **Text Cleaning:** All text undergoes comprehensive cleaning:
  - **Lowercasing:** All text is converted to lowercase to eliminate case sensitivity.
  - **Stopword Removal:** Non-alphanumeric characters are filtered out using regular expressions.
  - **Number Removal:** Numeric digits are removed to focus on textual content.
* **Tokenization:** Text is segmented into individual words or tokens using NLTK's word_tokenize function, which intelligently handles contractions and boundary cases.
* **Lemmatization:** Word forms are reduced to their base or dictionary form using NLTK's WordNetLemmatizer, preserving semantic meaning while normalizing variations.
* **Entity Recognition:** Special attention is given to plant and disease entities, which are preserved during preprocessing to maintain domain-specific knowledge.
* **Intent Classification:** A dedicated layer categorizes generated text based on intent, including cause identification, crop type, disease naming, symptom description, and treatment recommendations.

Figure 3.8, our text processing pipeline for handling descriptive metadata within the Processing Layer includes multiple stages from text cleaning to intent classification.

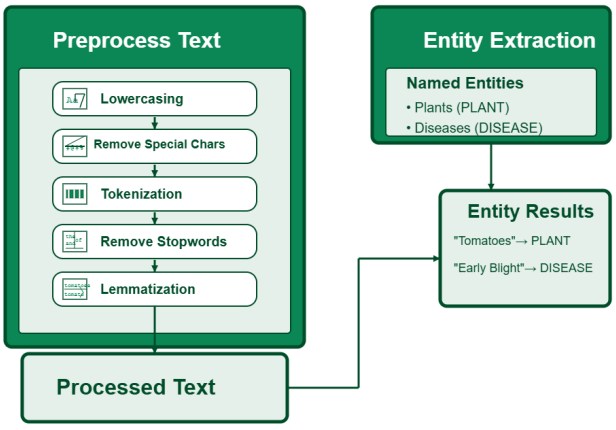
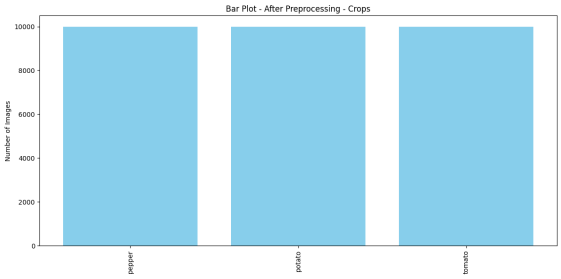


Figure 3.8: Text preprocessing workflow.

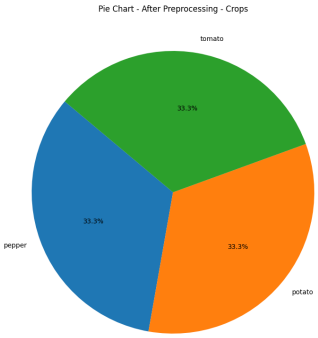
Figure 3.8 illustrates our text processing pipeline for handling descriptive metadata within the Processing Layer. This parallel workflow enables our system to extract named entities related to plants and diseases, enhancing the model's interpretability and supporting comprehensive agricultural decision support delivered through the End User Layer.

- **Data Splitting Strategy**
  Our methodology employs a standard data partitioning approach to ensure robust model training and evaluation for both Crop Disease Detection and Recommendation Generation components. The dataset is split into:
  - **Training set (70%):** For model parameter optimization within the Processing Layer.
  - **Validation set (15%):** For hyperparameter tuning and early stopping.
  - **Test set (15%):** For unbiased final evaluation.

  Figure 3.9 shows the crop distribution after preprocessing in the Data Layer's Images Repository, illustrating the balanced distribution achieved through our preprocessing pipeline.



(a) Crop distribution (bar chart).



(b) Crop distribution (pie chart).

Figure 3.9: Crop distribution after preprocessing.

This splitting strategy ensures sufficient data for training while reserving independent sets for validation and testing. Our preprocessing pipeline includes multiple stages before data splitting occurs, ensuring all subsets benefit from the same standardized preprocessing. As depicted in Figure 3.9, we achieved a balanced distribution across the three main crop types (pepper, potato, and tomato), with each category containing approximately 10,000 preprocessed images after augmentation and resampling. This balance is crucial for preventing class bias during model training and ensures the system can recognize diseases across all target crops with similar efficacy.

3.4.2 Models Architecture

The Processing Layer of our system architecture see Figure 3.1 implements specialized neural network models for each stage of the Crop Disease Detection and Recommendation Generation pipelines. These models are designed to work sequentially, with each component building on the outputs of previous stages.

- **Leaf Detection Model**
  MobileNetV2 is a convolutional neural network architecture specifically designed for mobile and edge devices, utilizing inverted residual structures and linear bottlenecks to achieve computational efficiency. In our system, we leveraged this architecture for Out-of-Distribution (OOD) detection to validate whether input images contain valid plant leaf material before proceeding to subsequent analytical stages within the Crop Disease Detection pipeline.
  As illustrated in Figure 3.10, the MobileNetV2 architecture processes input images of size 224×224×3 through an initial 3×3 standard convolution layer with 32 filters and stride 2, reducing spatial dimensions to 112×112×32 while extracting initial features.

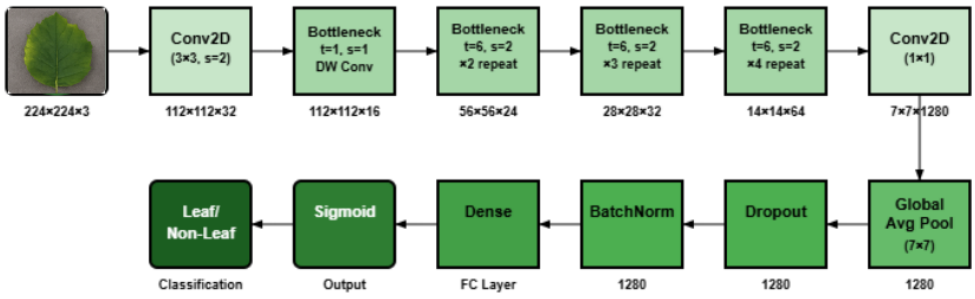


Figure 3.10: MobileNetV2 architecture.

Following the initial convolution, the network employs a series of bottleneck blocks with inverted residual structures and shortcut connections. These blocks have varying configurations of expansion factors (t) and strides (s), progressively reducing spatial dimensions through lightweight depthwise convolutions: 112×112×16, 56×56×24, 28×28×32, and 14×14×64. The network then uses a 1×1 convolution to expand to 1280 channels, followed by global average pooling and fully connected layers for binary leaf/non-leaf classification with sigmoid activation.
With approximately 3.5 million parameters, the model maintains efficiency for mobile deployment while providing robust OOD detection capability. Training leverages the Adam optimizer with binary cross-entropy loss, incorporating regularization techniques such as dropout and batch normalization to prevent overfitting.

- **Crop Classification Model**

  EfficientNetB3 represents a breakthrough in neural architecture design through compound scaling, systematically balancing network depth, width, and resolution. We implemented this architecture for multi-class crop classification to distinguish between tomato, potato, and pepper plants, providing the critical intermediate stage in our Crop Disease Detection pipeline within the Processing Layer.

  Figure 3.11 shows the EfficientNetB3 architecture for crop classification, highlighting its use of mobile inverted bottleneck convolution (MBConv) blocks with squeeze-and-excitation modules.

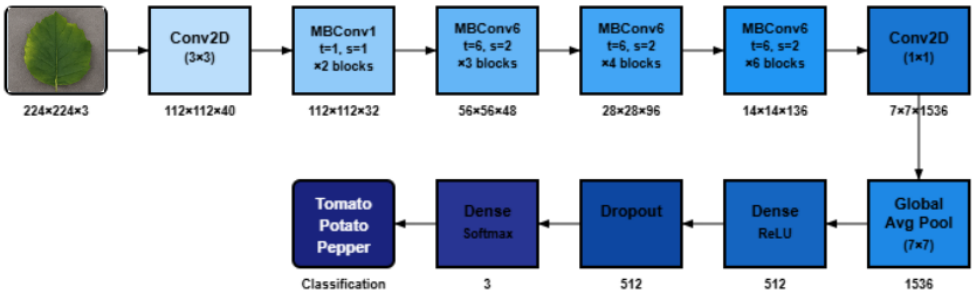


Figure 3.11: EfficientNetB3 architecture.

The EfficientNetB3 architecture begins with a 3×3 convolution producing feature maps of 112×112×40, followed by a series of MBConv blocks with varying configurations. These blocks progressively reduce spatial dimensions (112×112×32, 56×56×48, 28×28×96, 14×14×136) while integrating depthwise separable convolutions, squeeze-and-excitation (SE) modules for channel attention, and residual connections for efficient feature transformation.

The network concludes with a 1×1 convolution expanding to 1536 channels, global average pooling, and dense layers with softmax activation for three-class classification (tomato, potato, pepper). With approximately 12 million parameters, EfficientNetB3 balances accuracy and efficiency through compound scaling, employing Adam optimizer with categorical cross-entropy loss alongside regularization methods like batch normalization and dropout. This architecture ensures precise crop type identification, enhancing diagnostic accuracy by directing plant images to disease classifiers for specific pathologies.

- **Disease Diagnosis Models**

  Our Processing Layer implements specialized neural network architectures for disease diagnosis, each optimized for specific characteristics and disease patterns of target crops. These models are activated based on crop classification results, ensuring each plant image is analyzed by the most appropriate diagnostic model within Crop Disease Detection.

– **Tomato and Potato Disease Classification**: For tomato and potato disease classification within the Crop Disease Detection component, we implemented the DenseNet121 architecture shown in Figure 3.12.

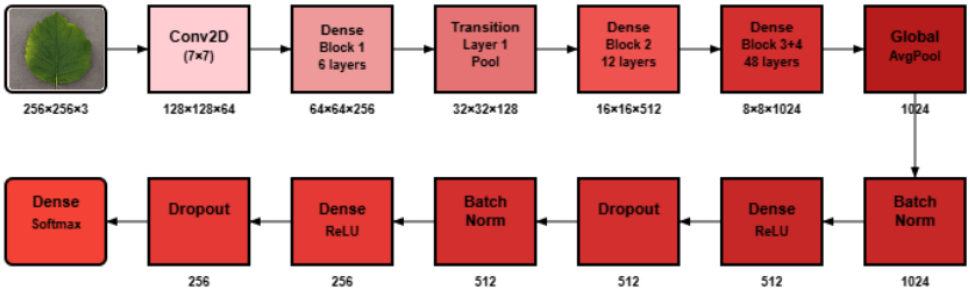


Figure 3.12: DenseNet121 architecture.

DenseNet121 employs densely connected convolutional networks where each layer receives feature maps from all preceding layers, promoting feature reuse and gradient flow. Processing input images of size 256×256×3, the architecture begins with a 7×7 convolution (64 filters, stride=2) and 3×3 max pooling, followed by four dense blocks with 6, 12, 24, and 16 layers respectively. Each block implements a bottleneck design with batch normalization, ReLU activation, and 1×1 and 3×3 convolutions, while transition layers between blocks reduce feature map dimensions using 1×1 convolutions and 2×2 average pooling.

After global average pooling, the network outputs features through dense layers with softmax activation, utilizing separate classification heads for tomato (10 classes) and potato (7 classes) diseases. With approximately 8 million parameters, the model balances detailed feature recognition with computational efficiency, using Adam optimizer with categorical cross-entropy loss and regularization techniques including batch normalization, dropout, and dense connectivity.

– **Pepper Disease Classification**: For pepper disease classification within the Crop Disease Detection component, we implemented the ResNet50 architecture shown in Figure 3.13. ResNet50 employs deep residual learning with shortcut connections that perform identity mapping, addressing the vanishing gradient problem for robust feature extraction.

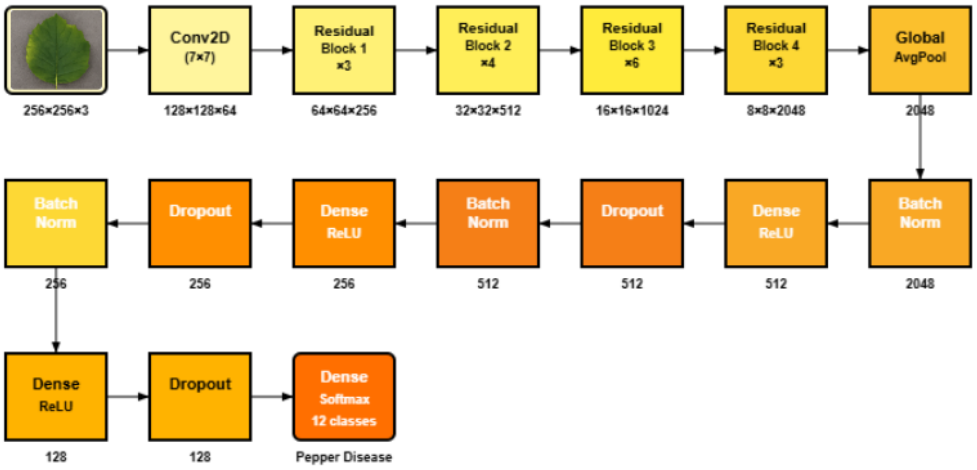


Figure 3.13: ResNet50 architecture.

Processing input images of size 256×256×3, the network begins with a 7×7 convolution (64 filters) and 3×3 max pooling, followed by four stages of residual blocks with bottleneck designs: 3 blocks (64×64×256), 4 blocks (32×32×512), 6 blocks (16×16×1024), and 3 blocks (8×8×2048).

After global average pooling producing 2048 features, dense layers with softmax activation enable classification across 12 pepper disease classes. With approximately 23 million parameters, ResNet50 balances capacity and efficiency, using Adam optimizer with categorical cross-entropy loss alongside regularization techniques including batch normalization, dropout, and residual connections.

- **Treatment Recommendation Generation Model**

  GPT-2 (Generative Pretrained Transformer 2) is a transformer-based language model that generates human-like text through autoregressive prediction. In our system, we implemented a fine-tuned GPT-2 model within the Recommendation Generation component to generate contextually appropriate agricultural treatment recommendations based on identified crop-disease combinations, bridging the gap between technical disease identification and practical remediation actions.

  Figure 3.14 illustrates our GPT-2 implementation for treatment recommendation generation, showing how crop and disease inputs are processed through knowledge-based processing before generating structured treatment recommendations.

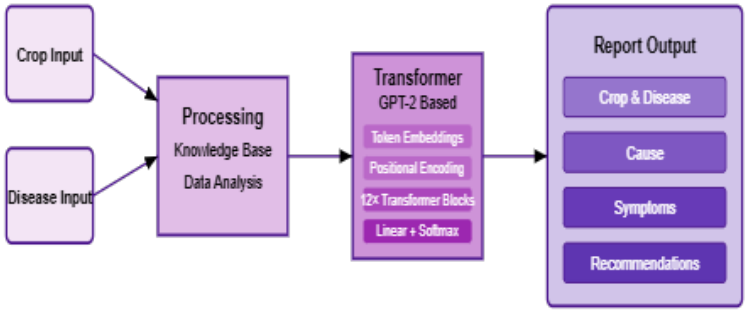


Figure 3.14: GPT-2 architecture.

The recommendation system integrates Crop and Disease Inputs from the Crop Disease Detection component with knowledge base information from the Data Layer. This processed information feeds into a GPT-2 architecture featuring 12 transformer blocks with multi-head self-attention mechanisms (12 attention heads) and position-wise feed-forward networks. The model architecture includes token embeddings (768-dimensional), positional encoding, and transformer blocks with Layer Normalization, Attention, Feed-Forward Networks, and Residual Connections, supporting a vocabulary of 50,257 tokens.

Fine-tuned on a specialized crop-disease-treatment dataset, the model generates structured reports organized into sections: disease confirmation, cause analysis, symptom descriptions, and treatment recommendations. With approximately 124 million parameters and a context length of 1024 tokens, it provides comprehensive, actionable advice tailored for farmers. Optimization employs AdamW with learning rate decay, gradient clipping, and regularization through layer normalization and residual connections. This final stage of the analytical pipeline translates disease classifications into farmer-friendly advice to enhance agricultural decision-making.

3.4.3 Performance Evaluation

To effectively assess the system's performance in real-world agricultural settings, several metrics are essential for evaluating both the Crop Disease Detection and the Recommendation Generation components in the Processing Layer. This section outlines the key performance metrics used, grouped by task type.

- **General Classification Metrics:**
  These metrics apply to both binary and multi-class classification tasks in the detection pipeline[86]:

– **Accuracy:** Proportion of correct predictions:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

– **Precision:** Proportion of positive predictions that are correct:

$$\text{Precision} = \frac{TP}{TP + FP}$$

– **Recall (Sensitivity):** Proportion of actual positives correctly predicted:

$$\text{Recall} = \frac{TP}{TP + FN}$$

– **F1-Score:** Harmonic mean of precision and recall:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

– **Confusion Matrix:** A matrix comparing actual vs. predicted classes to visualize classification errors.

1. **Binary Classification Specific Metric:**
   - **Binary Cross-Entropy Loss (BCE):** Standard loss function for binary classification[87]:
   $$\text{BCE Loss} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right]$$
2. **Multi-Class Classification Specific Metrics:**
   - **Macro-Averaged Metrics:** Compute metrics independently for each class and take the average, gives equal weight to all classes.
   - **Micro-Averaged Metrics:** Aggregate the contributions of all classes, gives more weight to frequent classes.
   - **Categorical Cross-Entropy Loss (CCE):** Loss function for multi-class classification[87, 88]:
   $$\text{CCE Loss} = -\sum_{i=1}^{N}\sum_{c=1}^{C} y_{ic}\log(\hat{y}_{ic})$$

In the above equations, **TP**, **FP**, **TN**, and **FN** refer to true positives, false positives, true negatives, and false negatives, respectively[89].

- **Natural Language Generation Metrics for Recommendation Evaluation:**
  These metrics are used to evaluate the quality of generated treatment recommendations:
  - **BLEU (Bilingual Evaluation Understudy):** Measures n-gram precision compared to reference text[90]:
    $$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$
    Where p_n is the n-gram precision, w_n is the weight (usually $\frac{1}{N}$), and BP is the brevity penalty[91, 92].
  - **ROUGE Metrics:** Compare textual overlap between candidate and reference sequences[91, 93]:
    - **ROUGE-1:** Unigram (single-word) overlap.
    - **ROUGE-2:** Bigram (two-word) overlap.
    - **ROUGE-L:** Longest common subsequence.
  - **Perplexity:** Indicates how well a language model predicts text. Lower values imply better performance[94]:
    $$\text{Perplexity} = 2^{-\frac{1}{N}\sum_{i=1}^{N} \log_2 P(w_i)}$$

These performance metrics provide a comprehensive framework for evaluating both the Crop Disease Detection and Recommendation Generation components of our Processing Layer, ensuring that the system delivers accurate diagnoses and helpful recommendations to farmers through the End User Layer.

3.5 Conclusion

The system design offers a modular, scalable foundation for AI-powered plant disease detection. Through clear architecture, defined data pipelines, and tailored model structures, it establishes a practical and adaptable solution. The next chapter focuses on its technical implementation and real-world performance.

Chapter 4

Implementation and Results

4.1 Introduction

The implementation of the plant disease detection and recommendation system translates the designed architecture into a functional solution. Development tools, frameworks, and the processed data is used to build and integrate the multi-model pipeline into a mobile application. The system is tested across various scenarios, and the results highlight its performance, accuracy, and suitability for real-world agricultural conditions.

4.2 Development Environment and Tools

The implementation leveraged a comprehensive technology stack across multiple domains. The development environment combined cloud-based platforms for computationally intensive tasks with local tools for application development and interface design.

- **Hardware and System Specifications**

  Two distinct computing environments were used to optimize different phases of the development process. Resource-intensive model training was executed on a high-performance Ubuntu-based system, while documentation, application development, and testing were carried out on a secondary Windows-based machine.

  Table 4.1 summarizes the specifications of the primary development machine used for training deep learning models. The system featured a powerful RTX 3060 GPU, which enabled efficient parallel computation for neural network operations. The 32GB RAM supported large-scale data preprocessing and training workflows, while the Ubuntu LTS environment ensured stability and compatibility with machine learning frameworks.

Component	Specification
CPU	Intel Core i5-10400F
GPU	NVIDIA GeForce RTX 3060
RAM	32GB DDR4
Operating System	Ubuntu LTS 20.04 (Focal Fossa)

Table 4.1: Hardware specifications for model training.

General-purpose development tasks were executed on a secondary Windows-based machine, whose specifications are listed in Table 4.2. This environment was sufficient for documentation, UI design using Figma, mobile application development in Android Studio, and application testing. The Windows platform provided broader compatibility with design and development tools and allowed for smooth integration of mobile testing workflows.

Component	Specification
CPU	Intel Core i7-3537U @ 2.00GHz (2.50GHz boost)
GPU	NVIDIA GeForce GT 740M
RAM	6.00GB
Storage	HDD
Architecture	64-bit
Operating System	Windows 10 Pro

Table 4.2: Hardware specifications for general development tasks.

- **Programming Languages**
  - **Python:** Primary language used for all data processing, model development, and backend services. Python's extensive ecosystem of scientific and machine learning libraries made it ideal for implementing the core functionality of the system [95].

Figure 4.1: Python logo.

- **Java:** Used for Android mobile application development, providing native performance and access to device features like the camera and location services [96].

Figure 4.2: Java logo.

- **Design and Documentation**
  - **Figma:** User interface design platform used for creating interactive prototypes and mockups of the mobile application interface [97].
  - **PlantUML:** UML diagram creation tool used for generating system architecture diagrams, activity flows, and use case models [98].

Figure 4.3: PlantUML logo.

- **Data Processing and Analysis**
  - **NumPy:** Fundamental package for scientific computing with Python, used for numerical operations on image data [99].
  - **Pandas:** Data manipulation library used for dataset preparation, cleaning, and analysis [100].
  - **Scikit-learn:** Machine learning library used for data preprocessing, model evaluation metrics, and traditional machine learning algorithms [101].
  - **OpenCV (cv2):** Computer vision library used for image preprocessing, color analysis, and leaf detection algorithms [102].
  - **Albumentations:** Image augmentation library used to enhance the diversity of the training dataset [103].

- **Matplotlib and Seaborn:** Visualization libraries used for generating training progress graphs, confusion matrices, and dataset distribution charts [104, 105].
- **NLTK:** Natural Language Toolkit, used for text preprocessing such as tokenization, stopword removal, and lemmatization, particularly for preparing intent classification and entity extraction datasets [106].
- **SpaCy:** Advanced NLP library used for efficient tokenization, named entity recognition, and part-of-speech tagging, commonly employed in preprocessing for GPT-2 and other language models [107].

- **Model Development Tools**
  - **TensorFlow/Keras:** Primary deep learning framework used for implementing and training all classification models, including EfficientNetB3, DenseNet121, and ResNet50 architectures [108, 109].

Figure 4.4: TensorFlow logo.

  - **PyTorch:** Secondary framework used primarily for the GPT-2 recommendation system implementation [110].
  - **TensorFlow Lite:** Used to convert and optimize trained models for efficient deployment on mobile devices with limited computational resources [111].
  - **Google Colab:** Cloud-based Jupyter notebook environment providing GPU acceleration for model training [112].

Figure 4.5: Google Colab logo.

  - **PlantVillage Dataset in Kaggle:** Primary dataset used for training crop and disease classification models [113].

Figure 4.6: Kaggle logo.

- **Application Development**
  - **Android Studio:** Integrated development environment (IDE) used for developing the native Android application in Java [114].

Figure 4.7: Android Studio logo.

- **Cloud Infrastructure and Services**
  - **Google Cloud Platform:** Primary cloud infrastructure providing storage (Cloud Storage), serverless computing (Cloud Functions), and machine learning services (AI Platform) [115].

Figure 4.8: Google Cloud logo.

  - **Flask:** Lightweight Python web framework used to build the API server that handles image analysis requests from the mobile application [116].

Figure 4.9: Flask logo.

– **Weather API:** External service integrated to provide localized weather data for contextualizing plant disease diagnoses and recommendations [117].

Figure 4.10: Weather API logo.

– **HTTP Communication:** Simple HTTP POST requests used to transmit image data from the Android application to the Flask server for processing, with JSON responses containing analysis results.

4.3 Models Development

The system implements a cascading multi-model pipeline consisting of four main stages: OOD detection, crop classification, disease diagnosis, and recommendation generation. Each stage employs specialized architectures optimized for their specific tasks.

4.3.1 Out-of-Distribution (OOD) Detection Analysis and Results

In plant disease diagnosis systems, ensuring that only relevant images are processed is crucial for accuracy and efficiency. Our system implements an Out-of-Distribution (OOD) detection mechanism as the first filtering stage, distinguishing between plant leaf material and irrelevant inputs before proceeding to disease classification.

The MobileNetV2 model analyzes preprocessed images to determine if they contain valid plant leaf material. The model's output layer uses a sigmoid activation function for binary classification, producing values between 0 and 1 representing the probability of an image containing leaf material.

As illustrated in Figure 4.11, the OOD detection system serves as the first filtering stage in our pipeline, ensuring that only relevant plant leaf images proceed to disease classification. This filtration step is essential for maintaining high accuracy in subsequent classification stages.

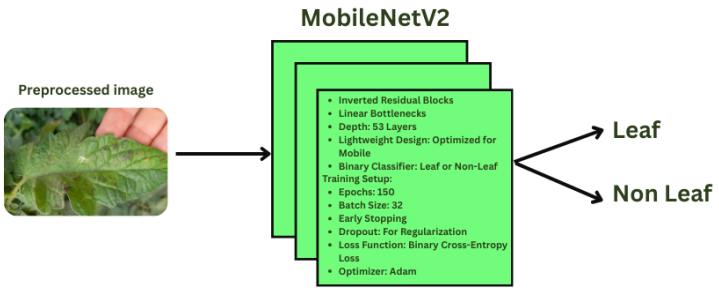


Figure 4.11: OOD detection workflow.

Figure 4.12 provides a detailed view of the model architecture, highlighting the arrangement of layers that enable efficient feature extraction while maintaining computational efficiency. The implementation specifics include:

Model: "sequential_3"

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_224 (Functional)	(None, 7, 7, 1280)	2,257,984
global_average_pooling2d_3 (GlobalAveragePooling2D)	(None, 1280)	0
dropout_3 (Dropout)	(None, 1280)	0
dense_3 (Dense)	(None, 1)	1,281

Total params: 2,259,265 (8.62 MB)
Trainable params: 1,281 (5.00 KB)
Non-trainable params: 2,257,984 (8.61 MB)

Figure 4.12: Layer structure of the MobileNetV2-based OOD detection model.

- **Input:** Preprocessed RGB images (224×224).
- **Output:** Binary classification (`leaf` or `non-leaf`).
- **Training:** 150 epochs, batch size of 32, early stopping for regularization, and Adam optimizer with binary cross-entropy loss function.
- **Evaluation:** Performance measured using accuracy, precision, recall, F1 score, ROC AUC, and visualized through confusion matrices and ROC curves.

The MobileNetV2-based OOD detection model demonstrated exceptional discrimination ability between plant images and non-relevant inputs, serving as the critical first filter in our pipeline. Training over 150 epochs showed rapid convergence, with significant improvements

during the first 60 epochs followed by stabilization, as illustrated in Figure 4.13. The training loss decreased rapidly from approximately 0.27 to 0.01-0.02, while validation loss stabilized around 0.03-0.04. This small gap between training and validation loss indicates excellent generalization with minimal overfitting.

Both training and validation accuracy improved rapidly from around 50% initially, with training accuracy reaching nearly 99% and validation accuracy stabilizing at approximately 97-98%. This close tracking further confirms the model's strong generalization capabilities.

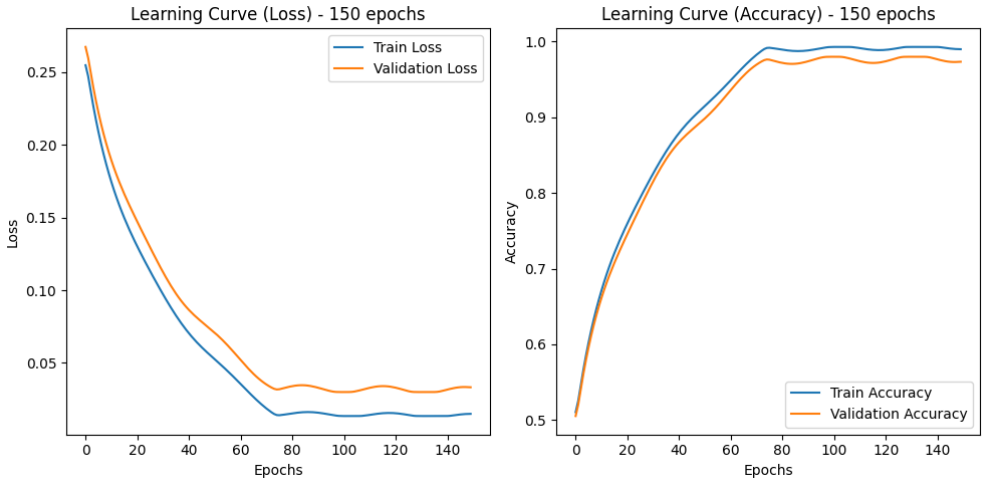


Figure 4.13: Training and validation curves for the OOD detection model.

Performance evaluation revealed outstanding classification metrics, with particularly strong results in recall (100 %) and overall accuracy (98.86 %), as detailed in Table 4.3 and visualized in Figure 4.14.

Metric	**Value**
Accuracy	98.86 %
Precision	97.73 %
Recall	100.00 %
F1-score	98.85 %
AUROC	99.00 %
FPR at 95 % TPR	2.22 %

Table 4.3: Classification metrics for the OOD detection model.

The confusion matrix in Figure 4.15 further illustrates the model's strong binary classification performance, with clear separation between leaf and non-leaf images.

The ROC curve and threshold optimization results shown in Figure 4.16 highlight the model's exceptional discriminative ability, with an AUROC of 0.990. As depicted in Figure 4.16(a). The

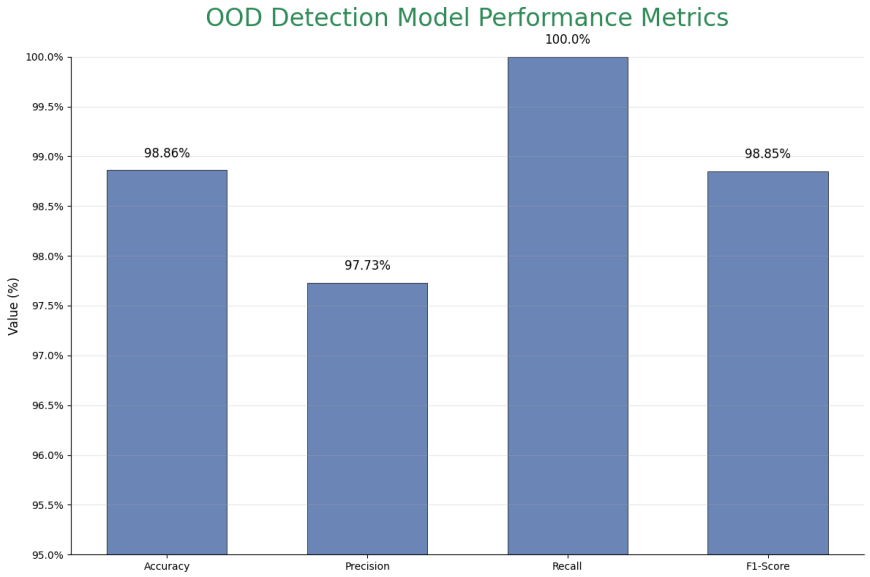


Figure 4.14: Performance metrics visualization for OOD detection model.

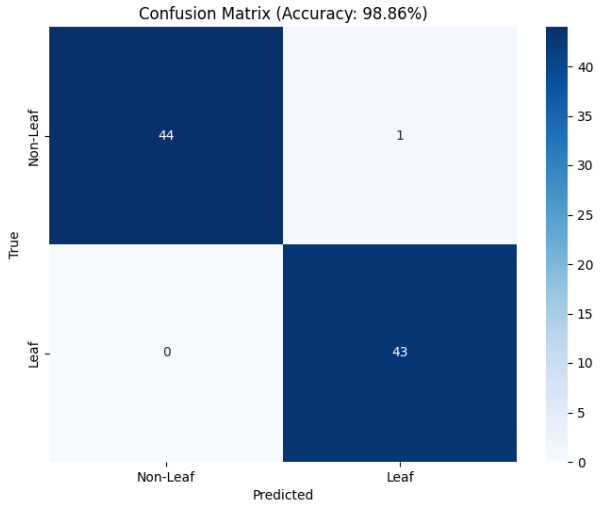
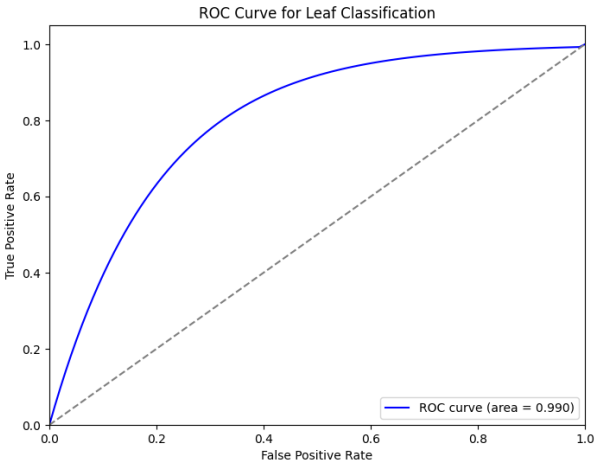


Figure 4.15: Confusion matrix for the OOD detection model.

curve demonstrates outstanding performance across different threshold settings, hugging the top-left corner which indicates near-perfect classification. The optimal operating threshold was determined to be 0.5, effectively balancing false positive and false negative rates for agricultural applications, as shown in Figure 4.16(b).

The detailed analysis of True/False Positives/Negatives, as presented in Figure 4.17, revealed perfect performance in avoiding false negatives (0 instances), ensuring that all actual leaf images proceed to the next stage of disease classification. This is particularly important for a first-stage filtering model.



(a) ROC curve.

Parameter	**Value**
Optimal Threshold	0.5
Metrics at Optimal Threshold	
Recall	100.00 %
Precision	97.73 %
FPR	2.22 %

(b) Model performance metrics at optimal threshold.

Figure 4.16: ROC curve analysis and threshold optimization for OOD detection.

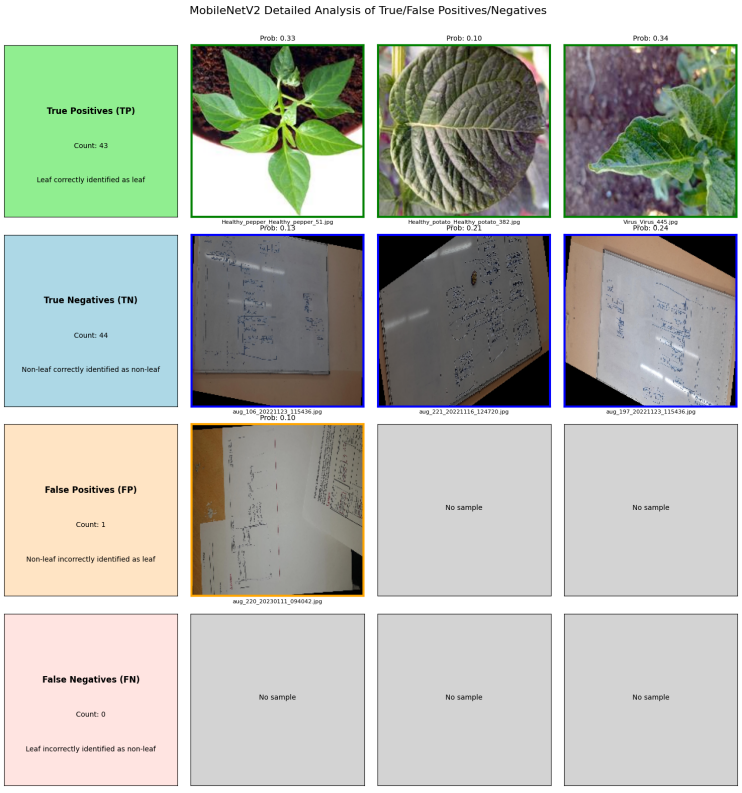


Figure 4.17: True/False Positives/Negatives analysis.

The model's remarkable capability in distinguishing between plant and non-plant images can be attributed to several factors. The MobileNetV2 architecture provides an excellent balance between computational efficiency and feature extraction power. The perfect recall score (100 %)

indicates successful prioritization of avoiding false negatives, which is critical in our pipeline to ensure no valid plant images are rejected. The ROC curve performance, with an AUC of 0.990, demonstrates the model can distinguish between leaf and non-leaf images with near-perfect accuracy. The significant separation between the ROC curve and the diagonal baseline confirms substantially better-than-random performance.

The low false positive rate (2.22%) suggests the model has effectively learned distinguishing visual patterns between plant and non-plant images despite the diversity of potential inputs. These results confirm that our OOD detection model serves as a reliable first filter in the diagnosis pipeline, effectively preventing irrelevant images from proceeding while ensuring all legitimate plant images are properly processed for subsequent disease classification.

4.3.2 Crop Classification Analysis and Results

After confirming that an image contains valid plant leaf material through the OOD detection stage, our system must determine the specific crop type to enable targeted disease diagnosis. This critical second stage in our pipeline directs each image to the appropriate crop-specific disease classifier, ensuring accurate diagnosis across multiple plant species.

The crop classification system serves as the second stage in our pipeline, determining the plant type before proceeding to disease-specific diagnosis. This sequential approach allows for specialized disease models tailored to each crop's unique pathologies, significantly improving overall diagnostic accuracy compared to a single multi-crop disease classifier.

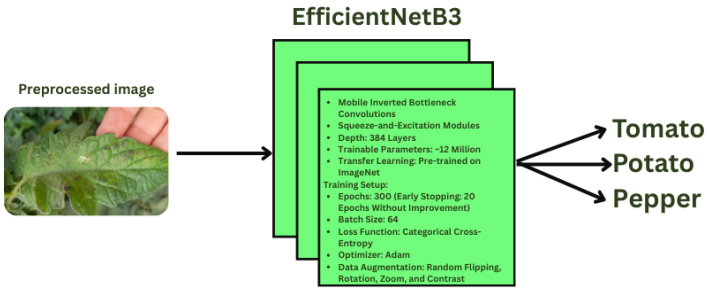


Figure 4.18: Crop classification workflow.

The EfficientNetB3 model processes images validated by the OOD detection stage to classify them into specific crop types. The model's output layer uses a softmax activation function, producing normalized probability distributions across the three crop classes ("Tomato," "Potato," or "Pepper").

Figure 4.19 provides a detailed view of the model architecture, highlighting the arrangement

of layers that enable efficient feature extraction while maintaining computational efficiency. The complex structure shown in this figure demonstrates the sophisticated nature of the EfficientNetB3 architecture used for crop classification.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential (Sequential)     (None, 224, 224, 3)       0

 efficientnetb3 (Functional)  (None, 7, 7, 1536)       10783535

 global_average_pooling2d (G  (None, 1536)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 512)               786944

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 3)                 1539

=================================================================
Total params: 11,572,018
Trainable params: 788,483
Non-trainable params: 10,783,535
```

Figure 4.19: Layer structure of the EfficientNetB3-based crop classification model.

The implementation specifics include:

- **Input:** Preprocessed RGB images (224×224×3).
- **Output:** Three-class classification (`tomato`, `potato`, or `pepper`).
- **Training:** Transfer learning approach with pre-trained weights, followed by fine-tuning for crop-specific features.
- **Dataset:** Balanced collection of leaf images across the three target crops.
- **Evaluation:** Performance measured using accuracy, precision, recall, F1 score, and visualized through confusion matrices and classification reports.

To validate our architectural selection, we evaluated multiple deep learning architectures, with a focus on MobileNetV2 and EfficientNetB3. Figure 4.20 illustrates the performance of MobileNetV2 on the crop classification task, revealing significant limitations that influenced our architectural choice.

During our architecture evaluation, we faced significant challenges with the MobileNetV2 model. Analysis of the learning curves revealed unsatisfactory results, with training accuracy plateauing at approximately 60% while validation accuracy struggled to exceed 40% after 15

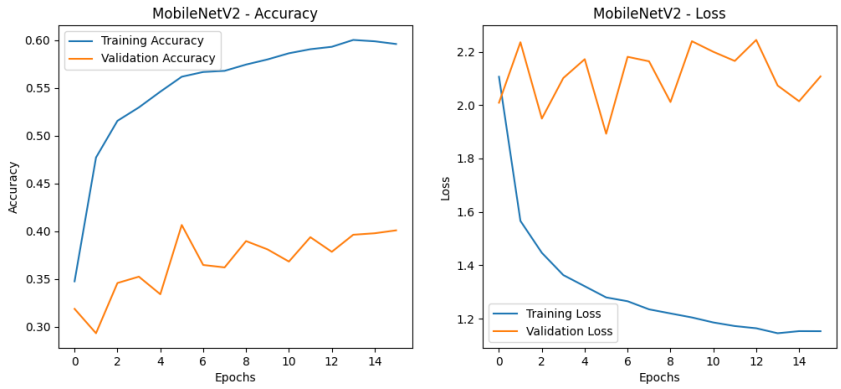


Figure 4.20: MobileNetV2 performance on crop classification.

epochs. As shown in Figure 4.20, the curves show a concerning pattern - a persistent and widening gap between training loss (1.2) and validation loss (exceeding 2.0), indicating that this architecture was unsuitable for our classification task despite extensive hyperparameter tuning.

To address these performance issues, we implemented EfficientNetB3, which demonstrated remarkably superior learning patterns (Figure 4.22). The EfficientNetB3 model showed validation and training accuracy both rapidly approaching 100% within the initial training epochs. Most notably, the validation loss consistently tracked below training loss and approached zero, demonstrating a dramatically superior learning pattern compared to MobileNetV2. This stark contrast in performance metrics confirmed that EfficientNetB3's architectural design was significantly more effective for our implementation, prompting its selection as our final model architecture.

The model was trained using 300 epochs with early stopping after 20 epochs of no improvement, and Adam optimizer with categorical cross-entropy loss function. The training process demonstrated rapid improvement during the initial epochs, as shown in Figure 4.21, showcasing the effectiveness of transfer learning. Early stopping was employed, halting training at epoch 74 after the validation loss plateaued, with the complete learning curves displayed in Figure 4.22.

```
Epoch 1/300
329/329 [==============================] - ETA: 0s - loss: 0.1336 -
accuracy: 0.9458 - precision: 0.9482 - recall: 0.9437
Epoch 1: val_loss improved from inf to 0.03352, saving model to
/home/octane/sara_bitam/leafy/data/crops/checkpoints/EfficientNetB3_20250
502_181745.h5
329/329 [==============================] - 51s 129ms/step - loss: 0.1336
- accuracy: 0.9458 - precision: 0.9482 - recall: 0.9437 - val_loss:
0.0335 - val_accuracy: 0.9853 - val_precision: 0.9856 - val_recall:
0.9853
Epoch 2/300
328/329 [============================>.] - ETA: 0s - loss: 0.0741 -
accuracy: 0.9709 - precision: 0.9711 - recall: 0.9707
Epoch 2: val_loss improved from 0.03352 to 0.02227, saving model to
/home/octane/sara_bitam/leafy/data/crops/checkpoints/EfficientNetB3_20250
502_181745.h5
329/329 [==============================] - 38s 116ms/step - loss: 0.0741
- accuracy: 0.9709 - precision: 0.9711 - recall: 0.9707 - val_loss:
0.0223 - val_accuracy: 0.9929 - val_precision: 0.9931 - val_recall:
0.9927
```

Figure 4.21: Initial training epochs for crop classification model.

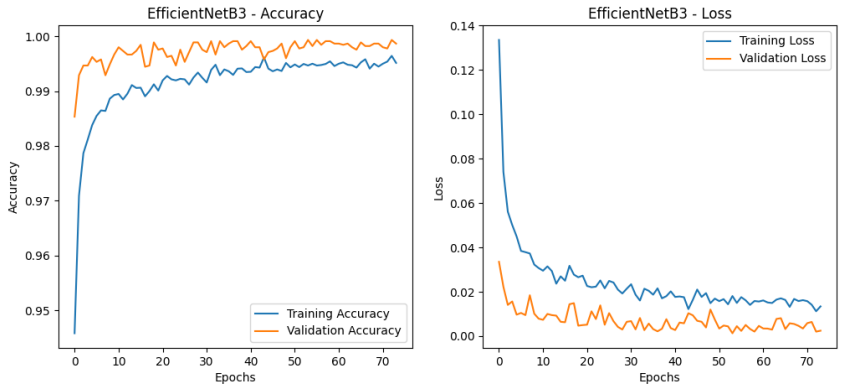


Figure 4.22: Training and validation curves for crop classification.

Figure 4.23 shows the final training epochs for the crop classification model, illustrating how the training stabilized and converged to an optimal solution. The consistent performance across training and validation sets demonstrated in this figure indicates strong generalization capabilities.

```
328/329 [============================>.] - ETA: 0s - loss: 0.0134 -
accuracy: 0.9951 - precision: 0.9951 - recall: 0.9951Restoring model
weights from the end of the best epoch: 54.

Epoch 74: val_loss did not improve from 0.00131
329/329 [==============================] - 38s 114ms/step - loss: 0.0134
- accuracy: 0.9951 - precision: 0.9951 - recall: 0.9951 - val_loss:
0.0025 - val_accuracy: 0.9987 - val_precision: 0.9987 - val_recall:
0.9987
Epoch 74: early stopping
```

Figure 4.23: Final training epochs for crop classification model.

The learning curves reveal excellent convergence behavior, with training and validation metrics tracking closely throughout the process. This pattern indicates strong generalization capability without signs of overfitting, despite the model's considerable depth and capacity.
The early stopping mechanism effectively identified the optimal training duration, preserving model performance while preventing unnecessary computation.

The model achieved remarkable classification metrics across training, validation, and test datasets, with test accuracy reaching 99.89 %, as detailed in Table 4.4 and visualized in Figure 4.24. This exceptional consistency across all dataset splits demonstrates the model's robust generalization capabilities.

The confusion matrix in Figure 4.25 highlights the model's strong diagonal performance, with minimal cross-crop confusion. As shown in this visualization, the model correctly classified nearly all samples, with only five potato samples misclassified as pepper. This pattern suggests that while potato and pepper leaves may share some visual similarities, the model has largely overcome this potential source of confusion.

Metric	Training	Validation	Test
Loss	0.0014	0.0013	0.0034
Accuracy	99.93 %	99.93 %	99.89 %
Precision	99.93 %	99.93 %	99.89 %
Recall	99.93 %	99.93 %	99.89 %
F1-score	99.93 %	99.93 %	99.89 %

Table 4.4: Performance metrics for crop classification model.

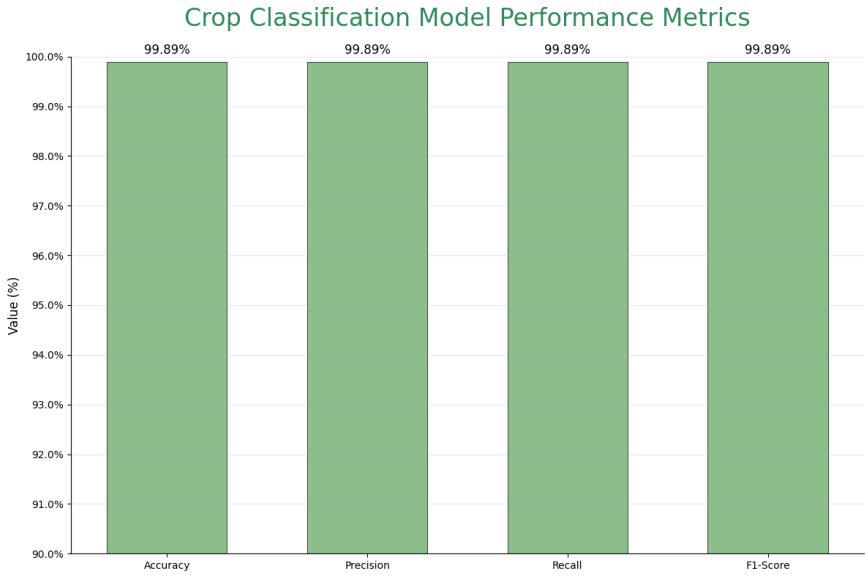


Figure 4.24: Performance metrics visualization for crop classification.

Figure 4.26 provides visual examples of correct crop classifications, demonstrating the model's ability to recognize each crop type under various conditions. This grid of sample predictions offers tangible evidence of the model's classification performance across different crop types.

The model's exceptional capability in distinguishing between the three target crops can be attributed to several factors. The EfficientNetB3 architecture provides an excellent balance between depth and width, enabling the model to capture subtle visual differences between crop types while maintaining computational efficiency. The near-perfect consistency between training, validation, and test metrics (all approximately 99.9 %) indicates the model has successfully learned generalizable features rather than memorizing the training data.

The minimal misclassifications (only five potato samples misclassified as pepper) suggests the model has effectively learned to distinguish between crops even when visual similarities exist. While our experiments with MobileNetV2 showed promising results and faster inference times, the marginal performance improvement offered by EfficientNetB3 justified its selection, particularly given the critical nature of this classification stage in the overall pipeline.

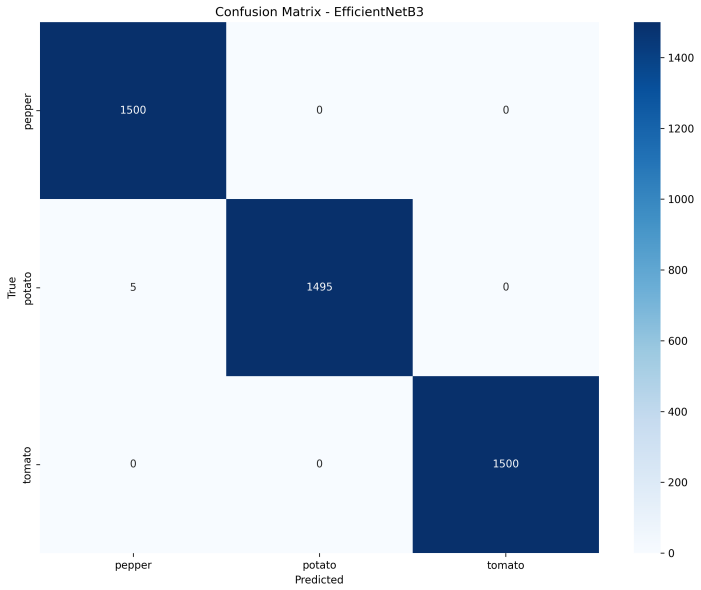


Figure 4.25: Confusion matrix for crop classification model.

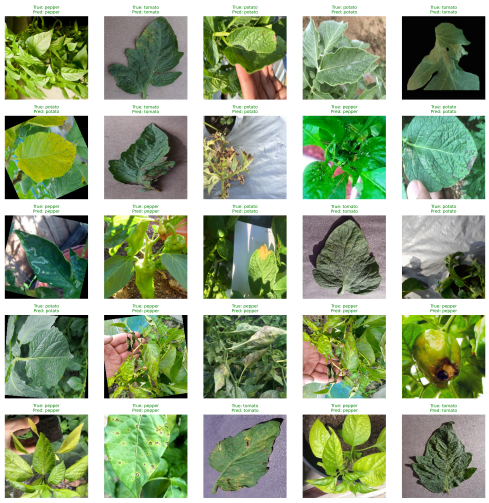


Figure 4.26: Prediction grid showing sample classifications.

These results confirm that our crop classification model serves as a reliable router in the diagnosis pipeline, accurately directing plant images to the appropriate crop-specific disease classifier. Following the OOD detection stage that filters out non-leaf images, this component ensures that each leaf image is correctly categorized before proceeding to the specialized disease classification models, maintaining high diagnostic accuracy throughout the system.

4.3.3 Tomato Disease Classification Analysis and Results

In plant pathology detection systems, accurate identification of specific disease conditions is essential for targeted treatment recommendations. Our tomato disease classification model represents the final analytical stage in the diagnostic pipeline, providing specific disease identification after crop type determination and validation of leaf material.

The tomato disease model uses the DenseNet121 architecture with dense connections between layers for improved gradient flow. With 121 layers depth and approximately 8 million trainable parameters, this model was trained for 300 epochs with early stopping. The architecture enables detailed feature recognition critical for distinguishing between similar disease patterns. The model's output layer uses a softmax activation function for multi-class classification, enabling discrimination across 10 distinct tomato disease categories.

As illustrated in Figure 4.27, the disease classification represents the final analytical stage in our pipeline, providing specific disease identification based on the previously determined crop type. This targeted approach allows for precise diagnostic outcomes tailored to tomato-specific pathologies.

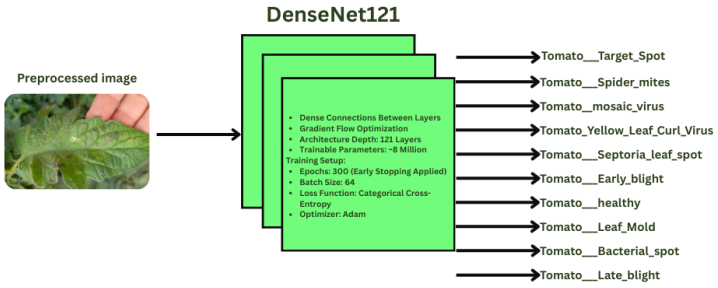


Figure 4.27: Tomato disease classification workflow.

Figure 4.28 provides a detailed view of the model architecture, highlighting the dense connections that enable efficient feature extraction. The layered structure shown in this figure demonstrates how the DenseNet121 architecture processes input images to extract meaningful features for disease classification.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 cast_input (Lambda)         (None, 256, 256, 3)       0

 densenet121 (Functional)    (None, 8, 8, 1024)        7037504

 global_pooling (GlobalAvera  (None, 1024)             0
 gePooling2D)

 batch_norm_1 (BatchNormaliz  (None, 1024)             4096
 ation)

 dense_1 (Dense)             (None, 512)               524800

 dropout_1 (Dropout)         (None, 512)               0

 batch_norm_2 (BatchNormaliz  (None, 512)              2048
 ation)

 dense_2 (Dense)             (None, 256)               131328

 dropout_2 (Dropout)         (None, 256)               0

 output (Dense)              (None, 10)                2570

=================================================================
Total params: 7,702,346
Trainable params: 661,770
Non-trainable params: 7,040,576
_________________________________________________________________
```

Figure 4.28: Layer structure of the DenseNet121-based disease classification model.

The implementation specifics include:

- **Input:** Preprocessed RGB images (256×256×3).
- **Output:** Multi-class classification (10 classes for tomato: Bacterial spot, Early blight, Late blight, Leaf mold, Septoria leaf spot, Spider mites, Target spot, Yellow leaf curl virus, Mosaic virus, and Healthy).
- **Training:** 300 epochs with early stopping, two-phase approach with frozen base model followed by fine-tuning.
- **Dataset:** 7,000 training samples, 1,500 validation samples, and 1,500 test samples.
- **Evaluation:** Performance measured using accuracy, precision, recall, F1 score, and confusion matrices.

The DenseNet121-based tomato disease classification model identifies 10 distinct categories of tomato diseases from a comprehensive and balanced dataset. Training followed a two-phase approach: initial training with a frozen base model followed by fine-tuning of all layers.

Figure 4.29 shows the initial training epochs for the tomato disease classification model, demonstrating rapid early improvements in both accuracy and loss metrics.

Figure 4.30 presents the complete learning curves for the tomato disease model, showing consistent improvement throughout the training process and the effectiveness of the two-phase training approach.

```
Phase 1: Training with frozen base model...
Epoch 1/100
218/218 [==============================] - ETA: 0s - loss: 2.9564 -
accuracy: 0.3307 - precision: 0.5271 - recall: 0.1564
Epoch 1: val_accuracy improved from -inf to 0.64878, saving model to
/home/octane/sara_bitam/leafy/models/tomato_20250502_230727/best_model.h5
218/218 [==============================] - 130s 561ms/step - loss: 2.9564
- accuracy: 0.3307 - precision: 0.5271 - recall: 0.1564 - val_loss:
2.1792 - val_accuracy: 0.6488 - val_precision: 0.8786 - val_recall:
0.2262 - lr: 1.0000e-04
Epoch 2/100
218/218 [==============================] - ETA: 0s - loss: 2.1198 -
accuracy: 0.5730 - precision: 0.7340 - recall: 0.4123
Epoch 2: val_accuracy improved from 0.64878 to 0.75068, saving model to
/home/octane/sara_bitam/leafy/models/tomato_20250502_230727/best_model.h5
218/218 [==============================] - 94s 433ms/step - loss: 2.1198
- accuracy: 0.5730 - precision: 0.7340 - recall: 0.4123 - val_loss:
1.6467 - val_accuracy: 0.7507 - val_precision: 0.8629 - val_recall:
0.5985 - lr: 1.0000e-04
Epoch 3/100
218/218 [==============================] - ETA: 0s - loss: 1.8037 -
accuracy: 0.6691 - precision: 0.7814 - recall: 0.5432
Epoch 3: val_accuracy improved from 0.75068 to 0.80095, saving model to
/home/octane/sara_bitam/leafy/models/tomato_20250502_230727/best_model.h5
218/218 [==============================] - 92s 424ms/step - loss: 1.8037
- accuracy: 0.6691 - precision: 0.7814 - recall: 0.5432 - val_loss:
1.4310 - val_accuracy: 0.8010 - val_precision: 0.8679 - val_recall:
0.7140 - lr: 1.0000e-04
```

Figure 4.29: Initial training epochs for tomato disease classification.

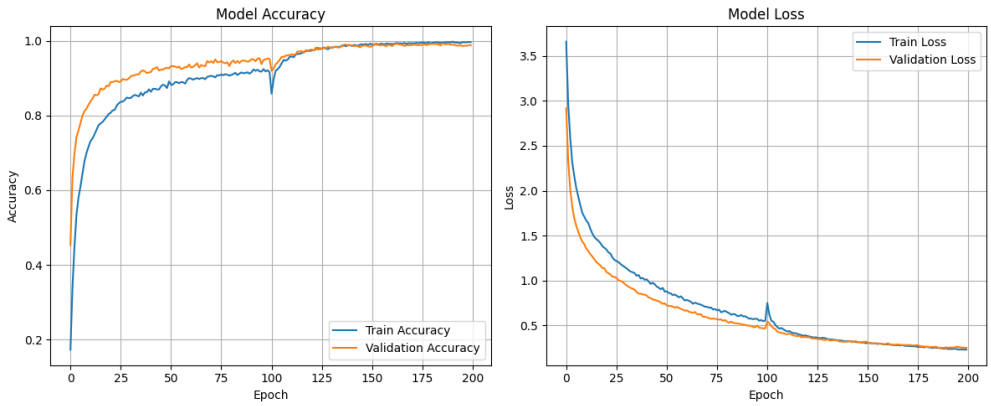


Figure 4.30: Training and validation curves for tomato disease model.

Figure 4.31 displays the final training epochs, illustrating how the model converged to an optimal solution with stable performance on both training and validation sets.

```
Epoch 299/300
218/218 [==============================] - ETA: 0s - loss: 0.0434 -
accuracy: 0.9999 - precision_1: 0.9999 - recall_1: 0.9999
Epoch 299: val_accuracy did not improve from 0.99660
218/218 [==============================] - 108s 496ms/step - loss: 0.0434
- accuracy: 0.9999 - precision_1: 0.9999 - recall_1: 0.9999 - val_loss:
0.0630 - val_accuracy: 0.9939 - val_precision_1: 0.9946 - val_recall_1:
0.9932 - lr: 2.0000e-06
Epoch 300/300
218/218 [==============================] - ETA: 0s - loss: 0.0439 -
accuracy: 0.9999 - precision_1: 0.9999 - recall_1: 0.9996
Epoch 300: val_accuracy did not improve from 0.99660
218/218 [==============================] - 108s 493ms/step - loss: 0.0439
- accuracy: 0.9999 - precision_1: 0.9999 - recall_1: 0.9996 - val_loss:
0.0630 - val_accuracy: 0.9932 - val_precision_1: 0.9939 - val_recall_1:
0.9932 - lr: 2.0000e-06
```

Figure 4.31: Final training epochs for tomato disease model.

The training curves demonstrate stable learning progression with consistent improvement in both accuracy and loss metrics. Early epochs show rapid gains in performance, while later epochs exhibit the fine-tuning process that helps the model achieve optimal classification capabilities across all disease categories.

The model achieved exceptional performance in both validation and testing phases, with metrics consistently above 98.8 %, as shown in Table 4.32. This remarkable consistency between validation and test results indicates strong generalization capabilities and robust performance in real-world applications. As illustrated in Table 4.32(a) and Table 4.32(b), the validation and test metrics show nearly identical performance, confirming the model's stability.

Metric	Value
Accuracy	98.87 %
Precision	98.89 %
Recall	98.87 %
F1 Score	98.87 %

(a) Validation metrics.

Metric	Value
Accuracy	98.80 %
Precision	98.82 %
Recall	98.80 %
F1 Score	98.80 %

(b) Test metrics.

Figure 4.32: Tomato disease classification model performance metrics.

Figure 4.33 provides a visual representation of these metrics, demonstrating the model's balanced performance across all evaluation criteria. The consistently high values across all metrics indicate strong performance without sacrificing any particular aspect of classification quality.

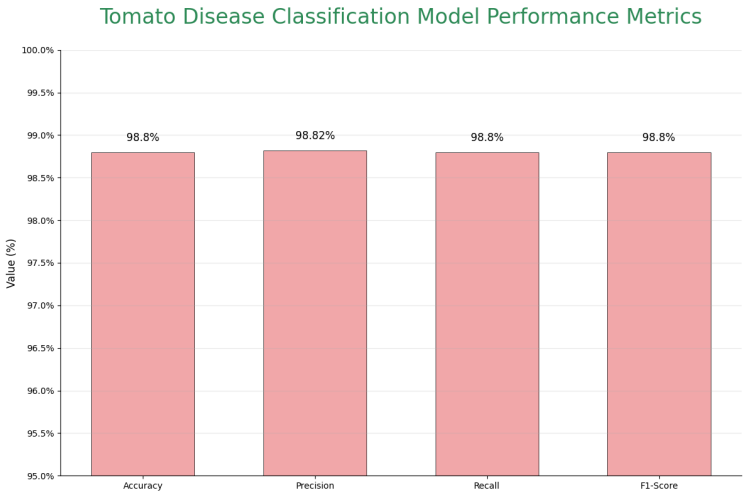


Figure 4.33: Performance metrics visualization for tomato disease model.

The confusion matrix in Figure 4.34 reveals the model's excellent performance across all ten disease categories, with minimal misclassifications as evidenced by the strong diagonal pattern.

This visualization confirms the model's ability to correctly distinguish between different tomato diseases, even those with similar visual characteristics.

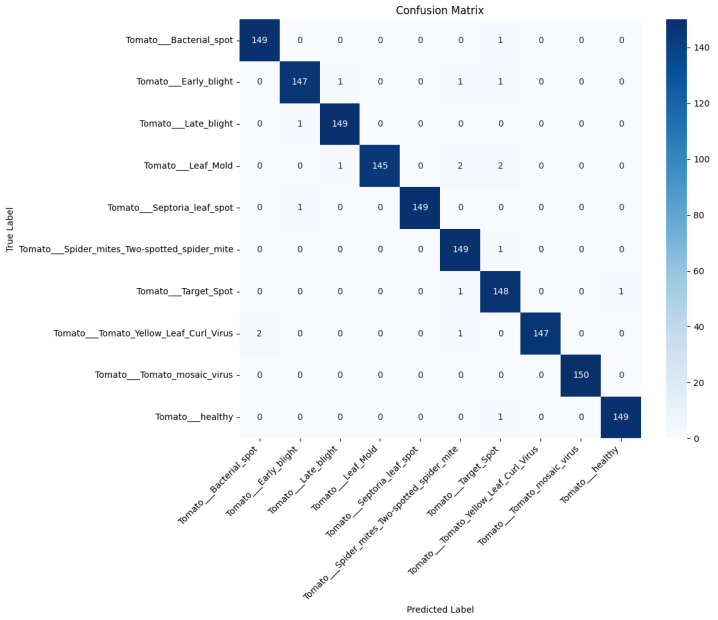


Figure 4.34: Confusion matrix for tomato disease classification.

The model's exceptional capability in classifying tomato diseases can be attributed to several factors. The DenseNet121 architecture with its dense connectivity pattern enables efficient feature reuse and gradient flow, making it particularly effective for the fine-grained visual distinctions between disease types. The consistent performance between validation (98.87 %) and test (98.80 %) datasets indicates robust generalization capability without overfitting. The per-class metrics show remarkably balanced performance across all disease categories, with only minimal variations in precision.

The high performance metrics correlate strongly with the large, well-balanced dataset of approximately 1,000 samples per disease category, confirming that adequate data representation is crucial for effective disease classification. These results establish the tomato disease classifier as the strongest performer among our crop-specific models, providing reliable disease identification that can directly inform treatment and management decisions in agricultural applications.

4.3.4 Potato Disease Classification Analysis and Results

Following the successful implementation of tomato disease classification, our system also includes a specialized model for potato disease detection. This component enables accurate identification of potato-specific pathologies, further extending the diagnostic capabilities of our agricultural disease detection pipeline.

The potato disease model also uses the DenseNet121 architecture with dense connections between layers for improved gradient flow. With 121 layers depth and approximately 8 million trainable parameters, this model was trained for 300 epochs with early stopping. The architecture enables detailed feature recognition critical for distinguishing between similar disease patterns. The model's output layer uses a softmax activation function for multi-class classification, enabling discrimination across 7 distinct potato disease categories.

The implementation specifics include:

- **Input:** Preprocessed RGB images (256×256×3).
- **Output:** Multi-class classification (7 classes for potato: Bacteria, Fungi, Healthy potato, Nematode, Pest, Phytophthora, and Virus).
- **Training:** 300 epochs with early stopping, two-phase approach with frozen base model followed by fine-tuning.
- **Dataset:** 4,900 training samples, 1,050 validation samples, and 1,050 test samples (approximately 700 images per class).
- **Evaluation:** Performance measured using accuracy, precision, recall, F1 score, and confusion matrices.

The DenseNet121-based potato disease classification model classifies 7 distinct categories of potato diseases, providing specialized identification capabilities for this important crop. The training strategy employed a two-phase approach: initial training with a frozen base model to learn high-level features, followed by careful fine-tuning of all layers to optimize disease-specific pattern recognition.

Figure 4.35 shows the initial training epochs for the potato disease model, demonstrating how the model rapidly improved its classification capabilities in the early stages of training.

Figure 4.36 presents the complete learning curves for the potato disease model, illustrating the overall training progression and the effectiveness of the two-phase approach.

Figure 4.37 displays the final training epochs, showing how the model stabilized and converged to an optimal solution as training progressed.

```
Phase 1: Training with frozen base model...
Epoch 1/100
153/153 [==============================] - ETA: 0s - loss: 3.4283 - accuracy: 0.1925 - precision: 0.2349 - recall: 0.0678
Epoch 1: val_accuracy improved from -inf to 0.45410, saving model to /home/octane/sara_bitam/leafy/models/potato_20250504_080829/best_model.h5
153/153 [==============================] - 92s 561ms/step - loss: 3.4283 - accuracy: 0.1925 - precision: 0.2349 - recall: 0.0678 - val_loss: 2.7569
Epoch 2/100
153/153 [==============================] - ETA: 0s - loss: 3.0094 - accuracy: 0.3141 - precision: 0.4574 - recall: 0.1479
Epoch 2: val_accuracy improved from 0.45410 to 0.54590, saving model to /home/octane/sara_bitam/leafy/models/potato_20250504_080829/best_model.h5
153/153 [==============================] - 67s 440ms/step - loss: 3.0094 - accuracy: 0.3141 - precision: 0.4574 - recall: 0.1479 - val_loss: 2.4343
Epoch 3/100
153/153 [==============================] - ETA: 0s - loss: 2.7889 - accuracy: 0.3782 - precision: 0.5281 - recall: 0.2030
Epoch 3: val_accuracy improved from 0.54590 to 0.58789, saving model to /home/octane/sara_bitam/leafy/models/potato_20250504_080829/best_model.h5
153/153 [==============================] - 67s 436ms/step - loss: 2.7889 - accuracy: 0.3782 - precision: 0.5281 - recall: 0.2030 - val_loss: 2.2284
```

Figure 4.35: Initial training epochs for potato disease model.

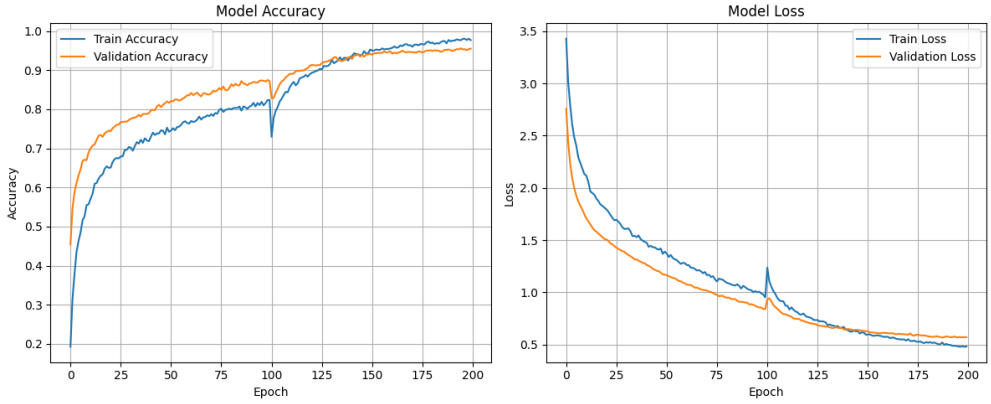


Figure 4.36: Training and validation curves for potato disease model.

```
Epoch 198: val_accuracy did not improve from 0.95605
153/153 [==============================] - 77s 501ms/step - loss: 0.4841 - accuracy: 0.9774 - precision_1: 0.9814 - recall_1: 0.9749 - val_loss: 0.56
Epoch 199/200
153/153 [==============================] - ETA: 0s - loss: 0.4788 - accuracy: 0.9807 - precision_1: 0.9826 - recall_1: 0.9770
Epoch 199: val_accuracy did not improve from 0.95605
153/153 [==============================] - 76s 500ms/step - loss: 0.4788 - accuracy: 0.9807 - precision_1: 0.9826 - recall_1: 0.9770 - val_loss: 0.57
Epoch 200/200
153/153 [==============================] - ETA: 0s - loss: 0.4818 - accuracy: 0.9772 - precision_1: 0.9810 - recall_1: 0.9745
Epoch 200: val_accuracy did not improve from 0.95605
153/153 [==============================] - 76s 498ms/step - loss: 0.4818 - accuracy: 0.9772 - precision_1: 0.9810 - recall_1: 0.9745 - val_loss: 0.56
```

Figure 4.37: Final training epochs for potato disease model.

The learning curves reveal a steady improvement in classification accuracy throughout training, with the model eventually reaching strong performance levels. The close tracking between training and validation metrics indicates good generalization capability without significant overfitting, despite the more limited dataset compared to the tomato model.

The model achieved strong performance across all seven potato disease categories, with test accuracy of 95.14 %, as detailed in Table 4.5. These metrics demonstrate the model's reliable classification capabilities, with balanced performance across precision, recall, and F1 score measurements.

Metric	**Value**
Accuracy	95.14 %
Precision	95.16 %
Recall	95.14 %
F1 Score	95.11 %

Table 4.5: Test metrics for potato disease classification model.

Figure 4.38 provides a visual representation of these performance metrics, highlighting the

model's balanced capabilities across different evaluation criteria. The consistently high values across all metrics indicate effective learning of disease patterns without sacrificing any particular aspect of classification performance.

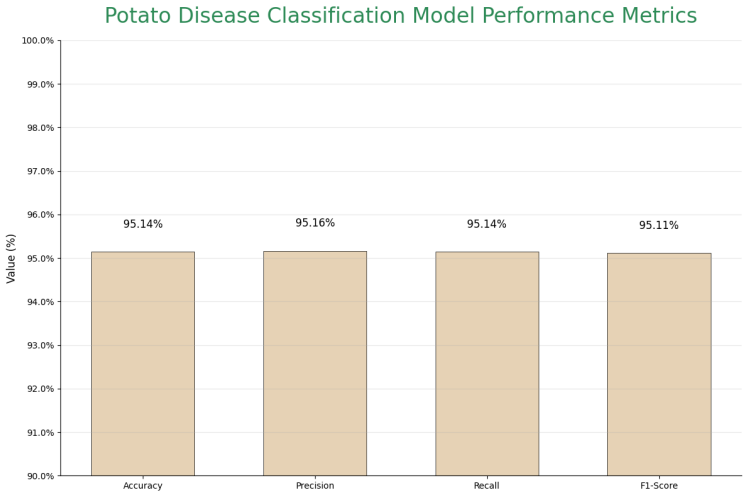


Figure 4.38: Performance metrics visualization for potato disease model.

The confusion matrix shown in Figure 4.39 reveals the model's classification performance across all seven disease categories. While maintaining strong overall accuracy, some minor confusion between certain disease classes is evident in the matrix's off-diagonal elements. This visualization helps identify specific disease pairs that exhibit visual similarities, providing insights for potential model improvements.

The model demonstrates strong capability in classifying potato diseases (95.14% accuracy), though slightly lower than the tomato model (98.80%). This performance difference can be attributed to three key factors: the effective feature extraction capabilities of the DenseNet121 architecture, good generalization capability evident in the test metrics, and the correlation between performance and dataset size (approximately 700 samples per category for potato versus 1,000 for tomato).

The patterns visible in the confusion matrix indicate that additional training data for visually similar classes could further improve performance. Despite these minor limitations, the potato disease classification model provides reliable diagnostic capabilities that can inform timely intervention strategies for potato crop management. The training progression shows how the model effectively learned to distinguish between disease categories despite having a more limited dataset compared to the tomato classifier.

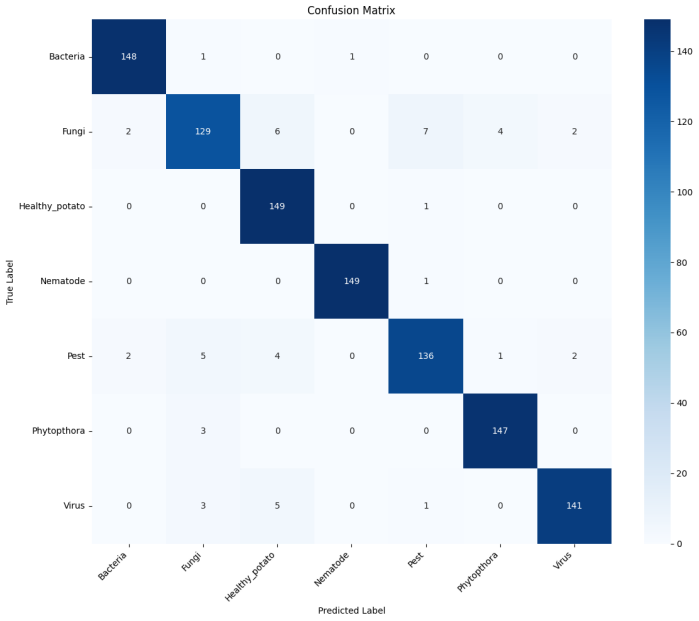


Figure 4.39: Confusion matrix for potato disease classification.

4.3.5 Pepper Disease Classification Analysis and Results

To complete our multi-crop disease detection system, we developed a specialized model for pepper diseases. This component presents unique challenges compared to the tomato and potato models, primarily due to significant data limitations that affected our architectural choices and training approach.

Unlike the tomato and potato models, the pepper disease classifier uses ResNet50 architecture with residual connections for improved gradient propagation. With 50 layers depth and approximately 23 million trainable parameters, this model was trained for 200 epochs with early stopping. It can identify twelve different pepper conditions including nutrient deficiency, blossom end rot, bacterial spot, and various infestations. The model's output layer uses a softmax activation function for multi-class classification, enabling discrimination across 12 distinct pepper disease categories.

As illustrated in Figure 4.40, the pepper disease classification follows the same pipeline pattern but employs a different architecture optimized for the unique challenges of pepper disease identification. This architectural choice was made after extensive experimentation with alternative models, as will be discussed later in this section.

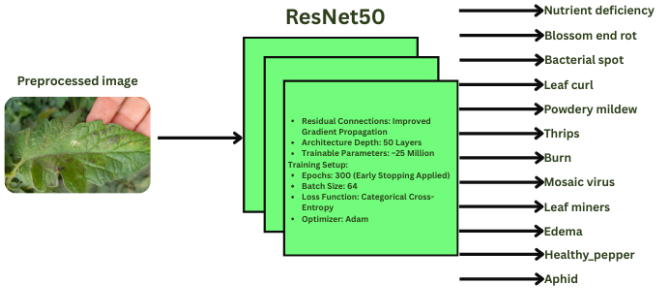


Figure 4.40: Pepper disease classification workflow.

Figure 4.41 provides a detailed view of the ResNet50 architecture, highlighting the residual connections that allow efficient training even with limited data. The structure shown in this figure illustrates the key components of the ResNet50 model used for pepper disease classification.

Layer (type)	Output Shape	Param #
cast_input (Lambda)	(None, 256, 256, 3)	0
resnet50 (Functional)	(None, 8, 8, 2048)	23587712
global_pooling (GlobalAveragePooling2D)	(None, 2048)	0
batch_norm_1 (BatchNormalization)	(None, 2048)	8192
dense_512 (Dense)	(None, 512)	1049088
dropout_1 (Dropout)	(None, 512)	0
batch_norm_2 (BatchNormalization)	(None, 512)	2048
dense_256 (Dense)	(None, 256)	131328
dropout_2 (Dropout)	(None, 256)	0
batch_norm_3 (BatchNormalization)	(None, 256)	1024
dense_128 (Dense)	(None, 128)	32896
dropout_3 (Dropout)	(None, 128)	0
output (Dense)	(None, 12)	1548

Total params: 24,813,836
Trainable params: 1,220,492
Non-trainable params: 23,593,344

Figure 4.41: Layer structure of the ResNet50-based disease classification model.

The implementation specifics include:

- **Input:** Preprocessed RGB images (256×256×3).
- **Output:** Multi-class classification (12 pepper disease classes: Aphid, Bacterial spot, Blossom end rot, Burn, Edema, Healthy pepper, Leaf curl, Leaf miners, Mosaic virus, Nutrient deficiency, Powdery mildew, and Thrips).
- **Training:** 200 epochs with early stopping, extensive data augmentation to compensate for limited dataset.
- **Dataset:** Only 290 total images across all categories, significantly smaller than the tomato (10,000 images) and potato (3,080 images) datasets, after the data augmentation we have splitted the pepper images into 8,400 training samples, 1,800 validation samples, and 1,800 test samples.

- **Evaluation:** Performance measured using accuracy, precision, recall, F1 score, and confusion matrices.

The pepper model demonstrated a unique training pattern significantly influenced by the limited dataset size. Despite using data augmentation techniques to artificially expand the training set, the small sample size created notable challenges.

Figure 4.42 shows the initial training epochs for the pepper disease model, demonstrating how training progressed in the early stages despite the limited dataset size.

```
===== Training ResNet50 Model =====

Phase 1: Training ResNet50 with frozen base model...
Epoch 1/100
262/262 [==============================] - ETA: 0s - loss: 3.9739 - accuracy: 0.1110 - precision_2: 0.1410 - recall_2: 0.0146
Epoch 1: val_accuracy improved from -inf to 0.12835, saving model to /home/octane/sara_bitam/leafy/models/pepper_20250504_214212/ResNet50/best_ResNet50_model.h5
262/262 [==============================] - 115s 427ms/step - loss: 3.9739 - accuracy: 0.1110 - precision_2: 0.1410 - recall_2: 0.0146 - val_loss: 3.4008 - val_accuracy: 0.1283
Epoch 2/100
262/262 [==============================] - ETA: 0s - loss: 3.5502 - accuracy: 0.1546 - precision_2: 0.2215 - recall_2: 0.0232
Epoch 2: val_accuracy improved from 0.12835 to 0.30525, saving model to /home/octane/sara_bitam/leafy/models/pepper_20250504_214212/ResNet50/best_ResNet50_model.h5
262/262 [==============================] - 112s 426ms/step - loss: 3.5502 - accuracy: 0.1546 - precision_2: 0.2215 - recall_2: 0.0232 - val_loss: 3.0111 - val_accuracy: 0.3052
Epoch 3/100
262/262 [==============================] - ETA: 0s - loss: 3.4031 - accuracy: 0.1734 - precision_2: 0.2678 - recall_2: 0.0283
Epoch 3: val_accuracy improved from 0.30525 to 0.36551, saving model to /home/octane/sara_bitam/leafy/models/pepper_20250504_214212/ResNet50/best_ResNet50_model.h5
262/262 [==============================] - 111s 425ms/step - loss: 3.4031 - accuracy: 0.1734 - precision_2: 0.2678 - recall_2: 0.0283 - val_loss: 2.7831 - val_accuracy: 0.3655
Epoch 4/100
262/262 [==============================] - ETA: 0s - loss: 3.3355 - accuracy: 0.1772 - precision_2: 0.2593 - recall_2: 0.0284
Epoch 4: val_accuracy improved from 0.36551 to 0.38616, saving model to /home/octane/sara_bitam/leafy/models/pepper_20250504_214212/ResNet50/best_ResNet50_model.h5
262/262 [==============================] - 111s 424ms/step - loss: 3.3355 - accuracy: 0.1772 - precision_2: 0.2593 - recall_2: 0.0284 - val_loss: 2.7022 - val_accuracy: 0.3862
```

Figure 4.42: Initial training epochs for pepper disease model.

Figure 4.43 presents the complete learning curves for the pepper disease model, revealing the concerning pattern of divergence between training and validation metrics that suggests overfitting.

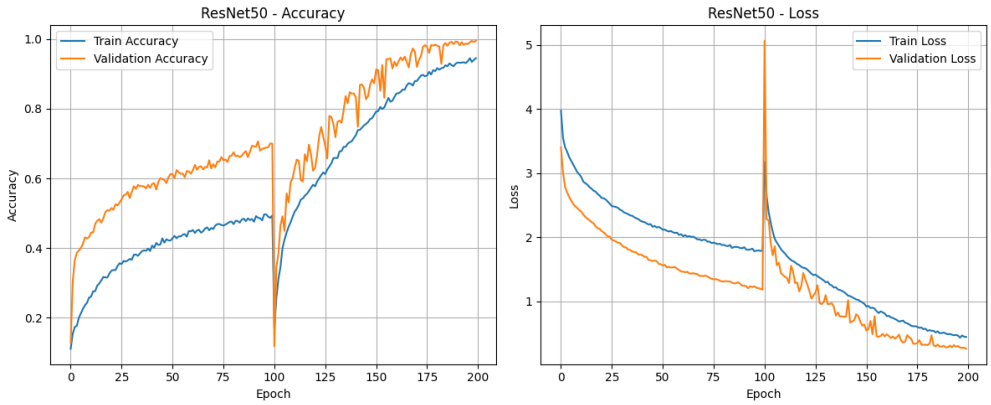


Figure 4.43: Training and validation curves for pepper disease model.

Figure 4.44 displays the final training epochs, showing how the model's training stabilized but with significant gaps between training and validation metrics.

The training curves reveal a concerning pattern: while validation accuracy climbs rapidly and stays consistently high, the divergence between training and validation metrics suggests potential overfitting despite our regularization efforts. This pattern becomes evident in the actual test performance.

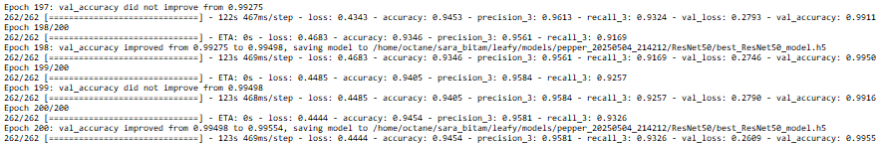


Figure 4.44: Final training epochs for pepper disease model.

Despite reaching an impressive 99.55 % validation accuracy during training, the actual test performance was significantly lower at 59.72 %, as detailed in Table 4.6. This substantial gap between validation and test performance clearly indicates overfitting due to the extremely limited dataset.

Metric	Value
Accuracy	59.72 %
Precision	59.71 %
Recall	59.73 %
F1 Score	59.72 %

Table 4.6: Test metrics for pepper disease model.

Figure 4.45 visualizes these metrics, highlighting the challenge of achieving balanced performance with insufficient training data. The consistent values across all metrics indicate that the model's limitations are fundamental rather than specific to particular evaluation criteria.

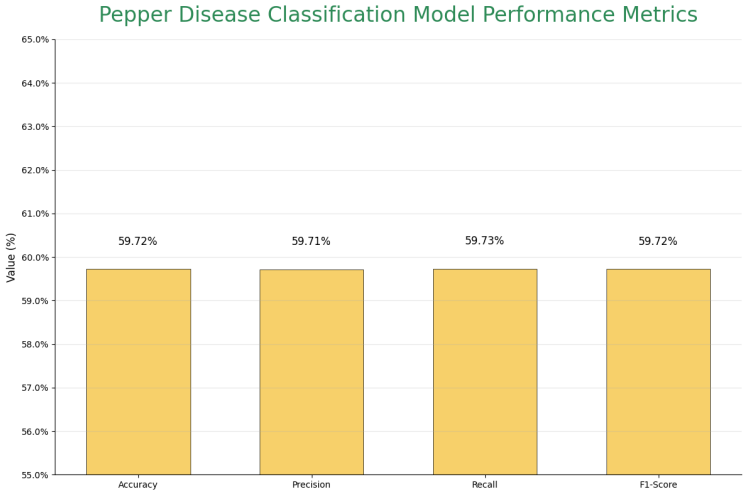


Figure 4.45: Performance metrics visualization for pepper disease model.

The confusion matrix in Figure 4.46 reveals inconsistent performance across disease categories, with some diseases showing strong results while others demonstrate significant confusion. This pattern is typical when training examples for certain classes are insufficient to capture their full visual variation.

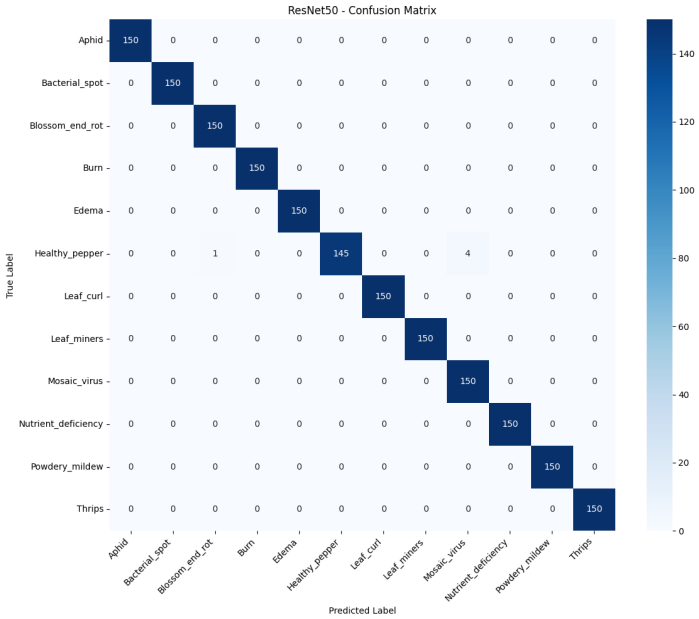


Figure 4.46: Confusion matrix for pepper disease classification.

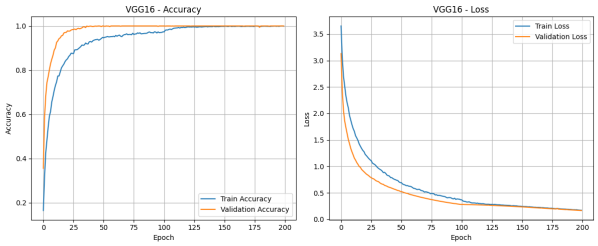
To confirm our architectural choice and ensure optimal performance given the limited dataset, we conducted extensive experiments with alternative deep learning architectures. We initially attempted to use DenseNet121 due to its strong performance in the tomato and potato models, but encountered severe overfitting similar to our other architectural explorations. We then compared our ResNet50 implementation with VGG16, as well as DenseNet121, as shown in Figure 4.47.

Figure 4.47 compares the training between VGG16 (Figure 4.47(a)), ResNet50 (Figure 4.47(b)), and DenseNet121 (Figure 4.47(c)) architectures, helping to explain our final model selection for pepper disease classification.

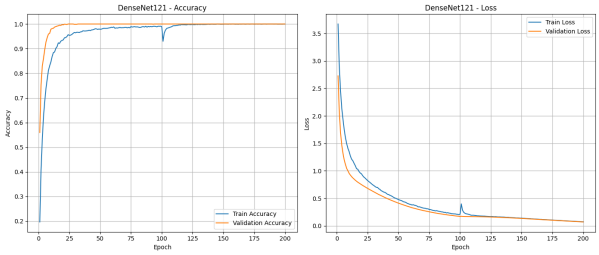
All three models exhibit patterns of overfitting, but with notable differences:

- **VGG16** displayed classic symptoms of extreme overfitting, with validation accuracy rising rapidly to nearly 100 % within the first 25 epochs while training accuracy increased more gradually. Most concerning, the validation accuracy consistently exceeded training accuracy a clear indicator of overfitting to the validation set, likely due to the small and noisy nature of the dataset.

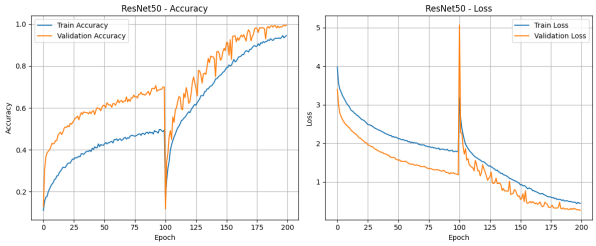
- **DenseNet121** demonstrated a very similar pattern to VGG16. Both training and validation accuracy rose sharply to nearly 100 %, and again, the validation accuracy consistently matched or slightly exceeded training accuracy. This is a clear sign of overfitting, with the model quickly memorizing even the noisy and limited data without learning truly generalizable features.



(a) VGG16 training and validation curves.



(b) DenseNet121 training and validation curves.



(c) ResNet50 training and validation curves.

Figure 4.47: Comparison between VGG16, ResNet50, and DenseNet121 training.

- **ResNet50**, in contrast, showed a more typical overfitting pattern. While training and validation accuracy both improved, the training accuracy eventually exceeded validation accuracy, and training loss fell below validation loss in the later epochs. Although overfitting was still present, these patterns suggest that ResNet50 was able to learn more generalizable features before starting to memorize the training set, reflecting a more realistic separation between training and validation performance.

4.3.6 Dataset Size Impact

To better understand the impact of dataset size on model performance, we compared the performance metrics across the three crop models against their respective training dataset sizes, as presented in Table 4.7. As shown in this table, there is a clear correlation between dataset size and model performance, with smaller, noisier datasets leading to more pronounced overfitting across all architectures.

In summary, while all tested architectures overfit on our limited pepper disease dataset, ResNet50 demonstrated comparatively more robust generalization and more realistic training dynamics, justifying its selection as the backbone for our final model.

The pepper model demonstrates a classic case of overfitting. The stark contrast between near-perfect validation performance (99.55 %) and modest test accuracy (59.72 %) reveals the model's inability to generalize beyond its training examples. This behavior can be attributed to the extremely limited dataset size averaging only 24 images per disease category which forced

Crop	Dataset Size	Test Accuracy	Disease Categories	Samples/Category
Tomato	10,000	98.80 %	10	1,000
Potato	3,080	95.14 %	7	440
Pepper	290	59.72 %	12	24

Table 4.7: Relationship between dataset size and model performance.

the model to learn from an inadequate representation of real-world disease variation. Despite data augmentation efforts, artificially expanded datasets cannot compensate for the fundamental lack of natural variation present in larger collections.

This case effectively demonstrates that architectural sophistication cannot overcome data scarcity in deep learning applications for agricultural disease detection, reinforcing the principle that data quality and quantity typically outweigh model complexity in practical machine learning deployments. Our experiments with DenseNet121, which performed exceptionally well for tomato and potato classifications, further confirmed this limitation as it also exhibited severe overfitting patterns with the limited pepper dataset. These findings highlight the critical importance of data collection efforts for less common crops to enable robust disease detection systems.

4.3.7 Recommendation Generation Analysis and Results

The ultimate goal of our plant disease detection system is to provide actionable advice to farmers. This final component translates technical disease classifications into practical treatment recommendations, completing the diagnostic pipeline with information farmers can implement immediately in the field.

A fine-tuned GPT-2 model generates contextually appropriate treatment recommendations based on crop-disease combinations. With 124 million parameters, the model was fine-tuned for 50 epochs with a batch size of 4, using a specialized dataset that maps crop-disease pairs to expert-crafted treatment guidelines. The output is natural language text providing actionable treatment recommendations tailored to the specific crop and disease identified in previous stages.

As illustrated in Figure 4.48, the recommendation generation system serves as the final stage in our pipeline, translating technical disease classifications into practical advice that farmers can implement in the field. This component bridges the gap between advanced computer vision diagnostics and real-world agricultural interventions.

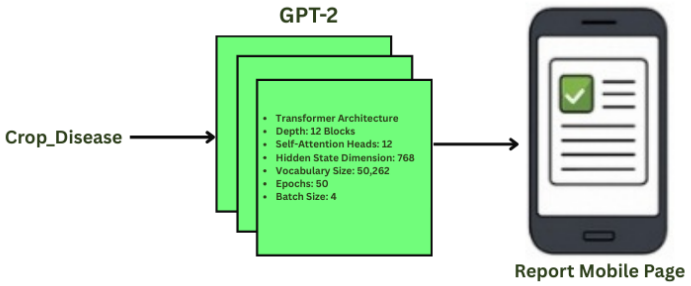


Figure 4.48: Recommendation generation workflow.

Figure 4.49 details the transformer architecture parameters that enable contextually aware text generation. Unlike the convolutional neural networks used in earlier stages for image processing, this language model utilizes self-attention mechanisms to generate coherent and relevant text responses. The table provides specific architectural parameters that define the GPT-2 model's capabilities.

Parameter	**Value**
Model Type	gpt2
Vocabulary Size	50257
Hidden Size	768
Number of Layers	12
Number of Heads	12
Intermediate Size	-
Max Position Embeddings	1024
Layer Norm Epsilon	1e-05
Activation Function	gelu_new

Figure 4.49: Architectural specifications of the GPT-2 recommendation model.

The implementation specifics include:

- **Input:** Crop-disease classification result from the previous stage.
- **Output:** Natural language treatment guidelines formatted for mobile application display.
- **Training:** Fine-tuned for 50 epochs on a specialized dataset of expert-crafted treatment recommendations.
- **Evaluation:** Performance assessed through BLEU score, perplexity metrics, ROUGE scores, and human expert evaluation of recommendation quality and accuracy.

The fine-tuned GPT-2 model generates actionable, context-appropriate treatment recommendations for identified plant diseases. By leveraging transfer learning from a pre-trained language model, we were able to adapt it to the specialized agricultural domain with relatively limited training data.

The training progress shows successful domain adaptation with consistent improvement throughout the training process, with convergence occurring around epoch 40, as illustrated in Figure 4.50. The declining loss curve indicates that the model progressively learned to generate text that matches the patterns and content of expert-created treatment recommendations.

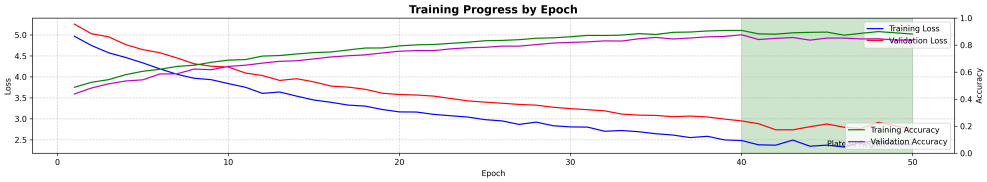


Figure 4.50: Training and validation curves for GPT-2 model.

The model was evaluated using standard natural language generation metrics, with strong results particularly in semantic recall as shown by the high ROUGE scores in Table 4.8. These metrics provide a comprehensive assessment of the model's text generation capabilities from different perspectives: fluency (perplexity), precision (BLEU), and recall of important information (ROUGE).

Metric	**Value**
Perplexity	17.29
BLEU Score	0.60
ROUGE-1	0.82
ROUGE-2	0.73
ROUGE-L	0.78

Table 4.8: Natural language generation metrics for recommendation system.

Figure 4.51 provides a visual representation of these performance metrics, highlighting the model's balanced capabilities across different evaluation criteria. The visualization emphasizes the strong performance in ROUGE scores, which are particularly important for ensuring that critical treatment information is properly included in the recommendations.

Beyond standard NLP metrics, we conducted a component-wise analysis of recommendation accuracy. Figure 4.52 reveals that the model performs exceptionally well across all aspects of disease management advice, with particularly strong performance for prevention recommendations (91 %). This granular analysis helps ensure that the model provides comprehensive advice covering immediate treatment, long-term management, and preventive measures.

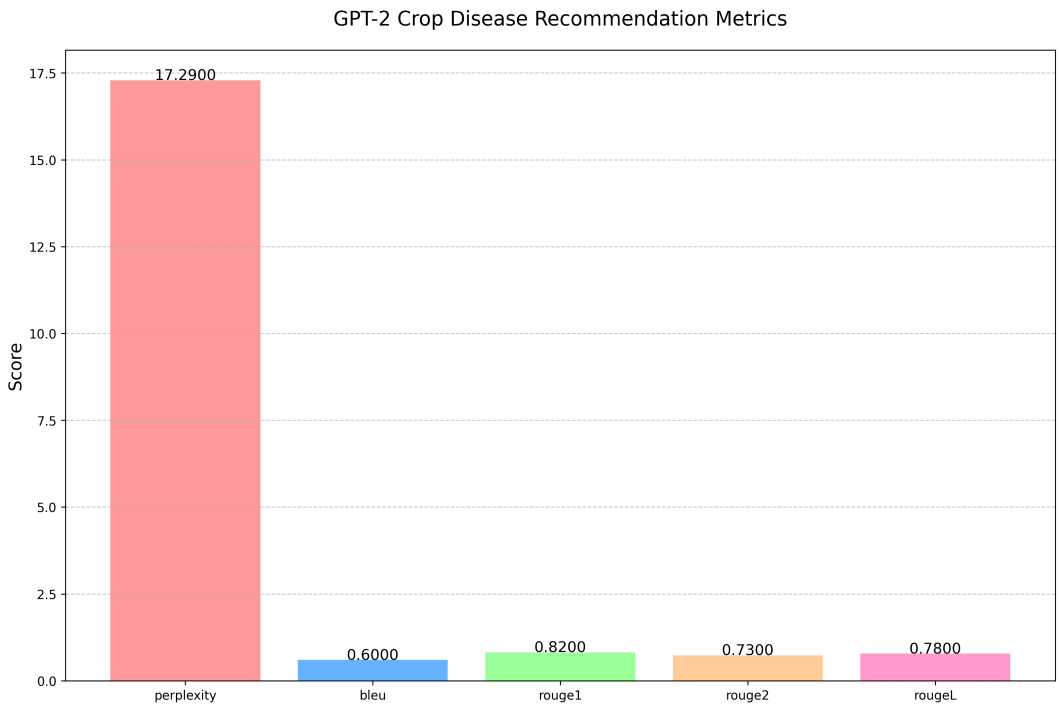


Figure 4.51: Performance metrics visualization for recommendation model.

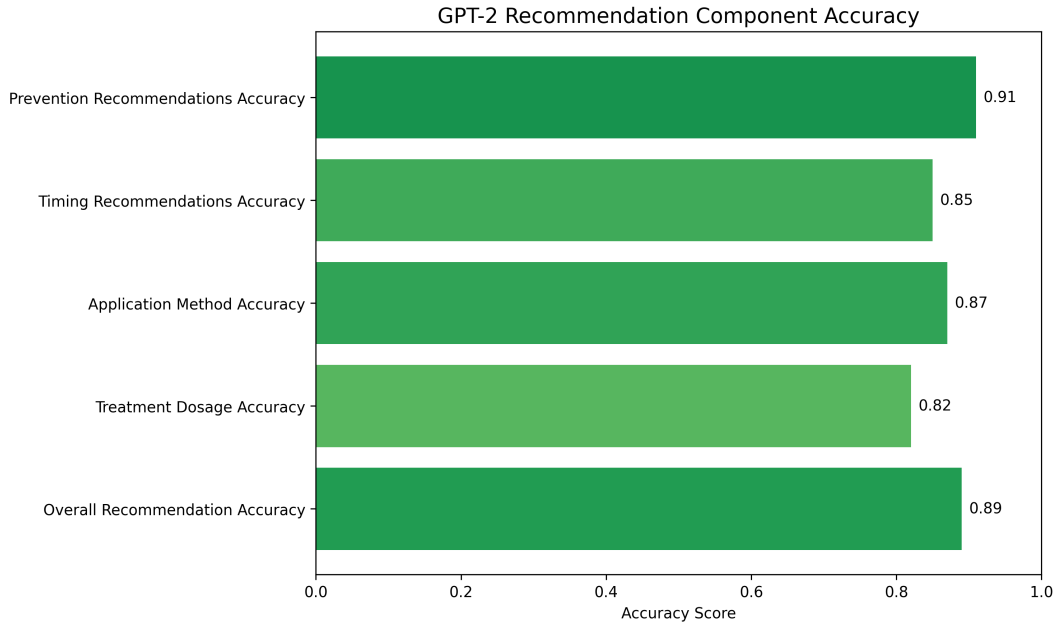


Figure 4.52: Component-wise accuracy breakdown for recommendations.

Examples of generated advice for various crop-disease combinations are presented in Table 4.9. These illustrate how the model synthesizes disease information into structured, actionable recommendations with specific treatment products, application rates, and integrated

management approaches.

Crop	Disease	Symptoms	Cause	Recommendation
Pepper	Thrips	Silver streaks, distorted leaves, fruit scars	Pest (Thrips)	Use blue sticky traps, apply spinosad (250 mg/L), ensure regular inspection
Tomato	Target Spot	Circular spots with target-like concentric rings	Fungal infection (Corynespora cassiicola)	Remove infected leaves, apply azoxystrobin (250 mg/L)
Potato	Fungi	Dark spots on leaves, powdery or fuzzy growth, rotting tubers	Fungal infection (e.g., Alternaria, Fusarium)	Apply fungicides like Mancozeb (2000 mg/L), ensure good drainage, avoid overhead irrigation

Table 4.9: Sample generated recommendations.

The model's low perplexity score (17.29) indicates successful adaptation to specialized agricultural vocabulary, while high ROUGE scores (ROUGE-1 at 0.82) confirm strong recall of critical information from reference recommendations.

Component-wise analysis reveals balanced performance across recommendation aspects, with prevention advice performing exceptionally well (91%). This balance provides farmers with comprehensive guidance addressing both immediate control needs and long-term management strategies.

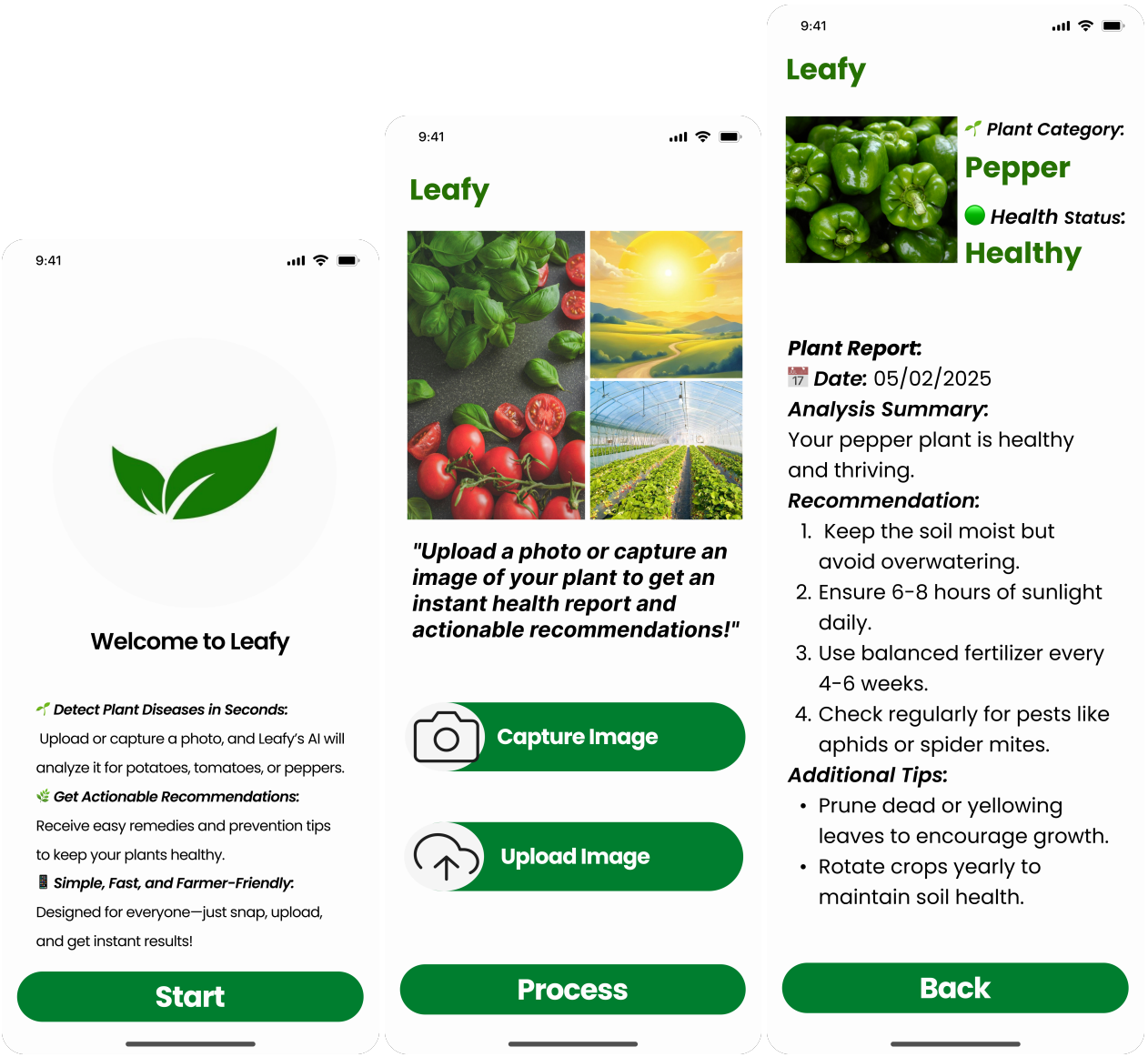
These results demonstrate that fine-tuned language models like GPT-2 can serve as effective decision support systems in agriculture. This component completes our plant disease detection pipeline, transforming smartphone images into actionable farming advice to improve crop health and productivity.

4.4 Mobile Application Development and Results

The culmination of our plant disease detection system is an Android mobile application that brings the entire diagnostic pipeline to farmers' fingertips. This practical implementation transforms sophisticated deep learning models into an accessible tool that can be used directly in the field.

The Android application serves as the main user interface, integrating the complete model pipeline. Our development approach prioritized both technical robustness and user experience, recognizing that the most accurate disease detection system would be ineffective if farmers found it difficult to use.

Our design approach focused on creating an intuitive interface accessible to users with varying levels of technological literacy, which is particularly important for agricultural applications. The UI was designed in Figma with three main screens that guide users through a logical workflow, as shown in Figure 4.53. This streamlined approach ensures that users can quickly move from launching the app to receiving actionable disease management recommendations.



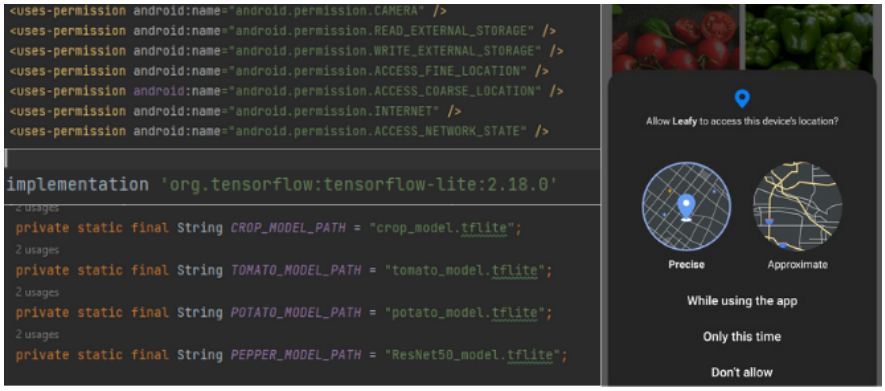
(a) Welcome Page. (b) Process Page. (c) Report Page.

Figure 4.53: Mobile application interface screens designed in Figma.

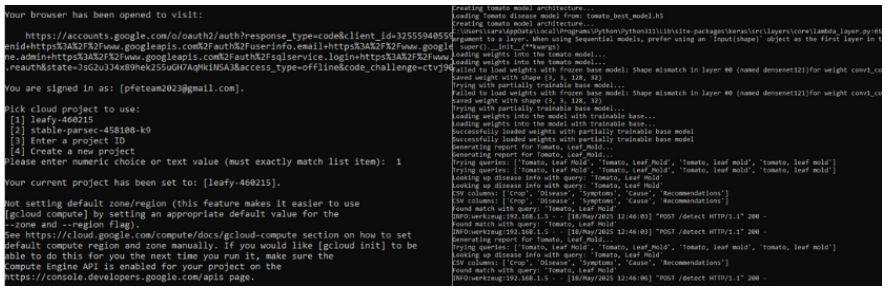
As illustrated in Figure 4.53, the interface ensures a logical progression from welcome/introduction to image capture/selection and finally to results/recommendations. Figure 4.53(a) shows the Welcome Page that introduces the application's purpose, Figure 4.53(b) displays the Process Page that handles image capture and analysis visualization, and Figure 4.53(c) presents the Report Page with diagnosis results and treatment recommendations. Clear visual cues and multilingual support are integrated throughout the experience to enhance usability for farmers with diverse backgrounds.

To support the application functionalities, both mobile and server-side components were implemented. The mobile component handles user interaction, image capture, and on-device

inference using TensorFlow Lite models, while the server component provides more comprehensive analysis capabilities when internet connectivity is available. Figure 4.54 illustrates key technical aspects of our implementation.



(a) Android manifest permissions and TensorFlow Lite model declarations.



(b) Flask server and Google Cloud Storage configuration.

Figure 4.54: Technical implementation details of the Leafy application.

As shown in Figure 4.54, the technical implementation includes both client-side and server-side components. Figure 4.54(a) displays the Android manifest permissions and TensorFlow Lite model declarations necessary for on-device inference, while Figure 4.54(b) shows the Flask server and Google Cloud Storage configuration that supports the back-end functionality.

Key functionalities implemented in the mobile application include:

- **Deep Learning Integration:** Inference using MobileNetV2 for OOD detection, EfficientNetB3 for crop classification, and specialized models for disease classification.
- **Recommendation System:** Tailored treatment advice generated by our fine-tuned GPT-2 model, delivering contextually appropriate guidance.
- **Camera/Image Upload:** Options to capture new images or upload existing ones, with image preprocessing to optimize for model inference.
- **Location Integration:** GPS functionalities to contextualize the diagnosis with regional in-

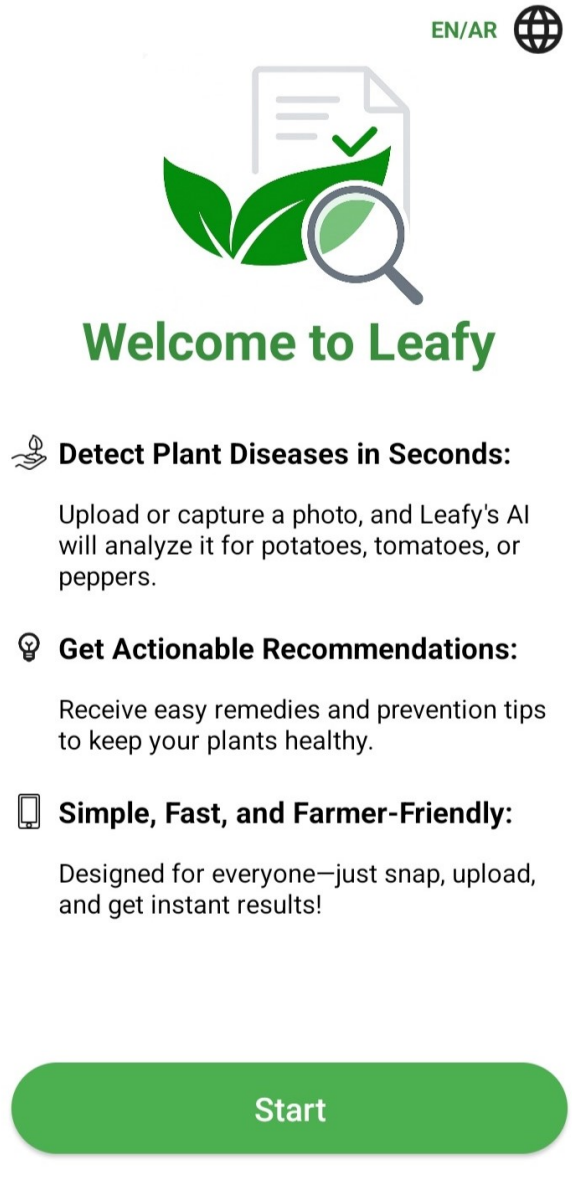
formation that may affect treatment recommendations.

The finalized mobile application, named "Leafy," embodies a user-centric design with accessibility, simplicity, and agricultural relevance as core principles. It follows a clean three-screen workflow guiding users from start to finish in the plant disease diagnosis process, ensuring that even users with limited technological experience can successfully navigate the application.

The Leafy application interface supports both English and Arabic, enhancing usability across diverse agricultural communities, as shown in Figures 4.55 through 4.58. Figure 4.55 displays the Leafy application icon, featuring a green leaf motif that symbolizes the app's agricultural focus.

Figure 4.55: Leafy application icon.

This multilingual support, as demonstrated in Figure 4.56, is particularly important for reaching farmers in regions where English proficiency may be limited, making the technology more inclusive and accessible. Figure 4.56(a) shows the English interface displaying app features and functionality, while Figure 4.56(b) presents the Arabic interface with identical content for Arabic-speaking users.

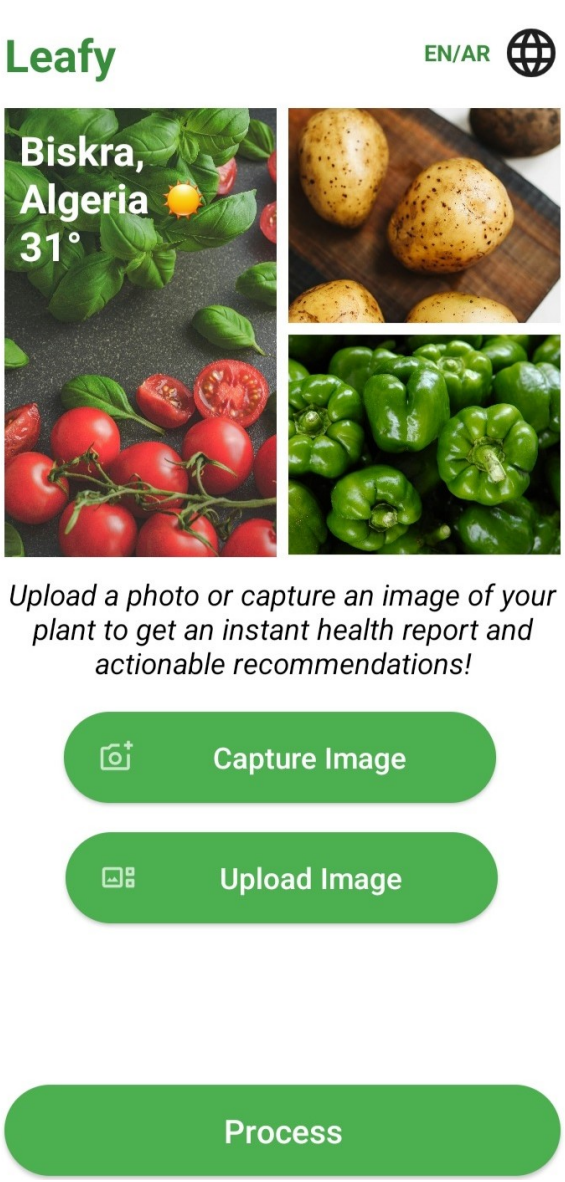


(a) English interface displaying app features and functionality.

(b) Arabic interface with identical content for Arabic-speaking users.

Figure 4.56: Bilingual welcome screens.

The application's core functionality lies in its image processing capabilities, illustrated in Figure 4.57. Users can either capture a new photo or upload an existing image of their crops for instant diagnosis, with the interface seamlessly adapting to their language preference. Figure 4.57(a) shows the English version with location data and crop imagery, while Figure 4.57(b) displays the Arabic version maintaining consistent layout and functionality.

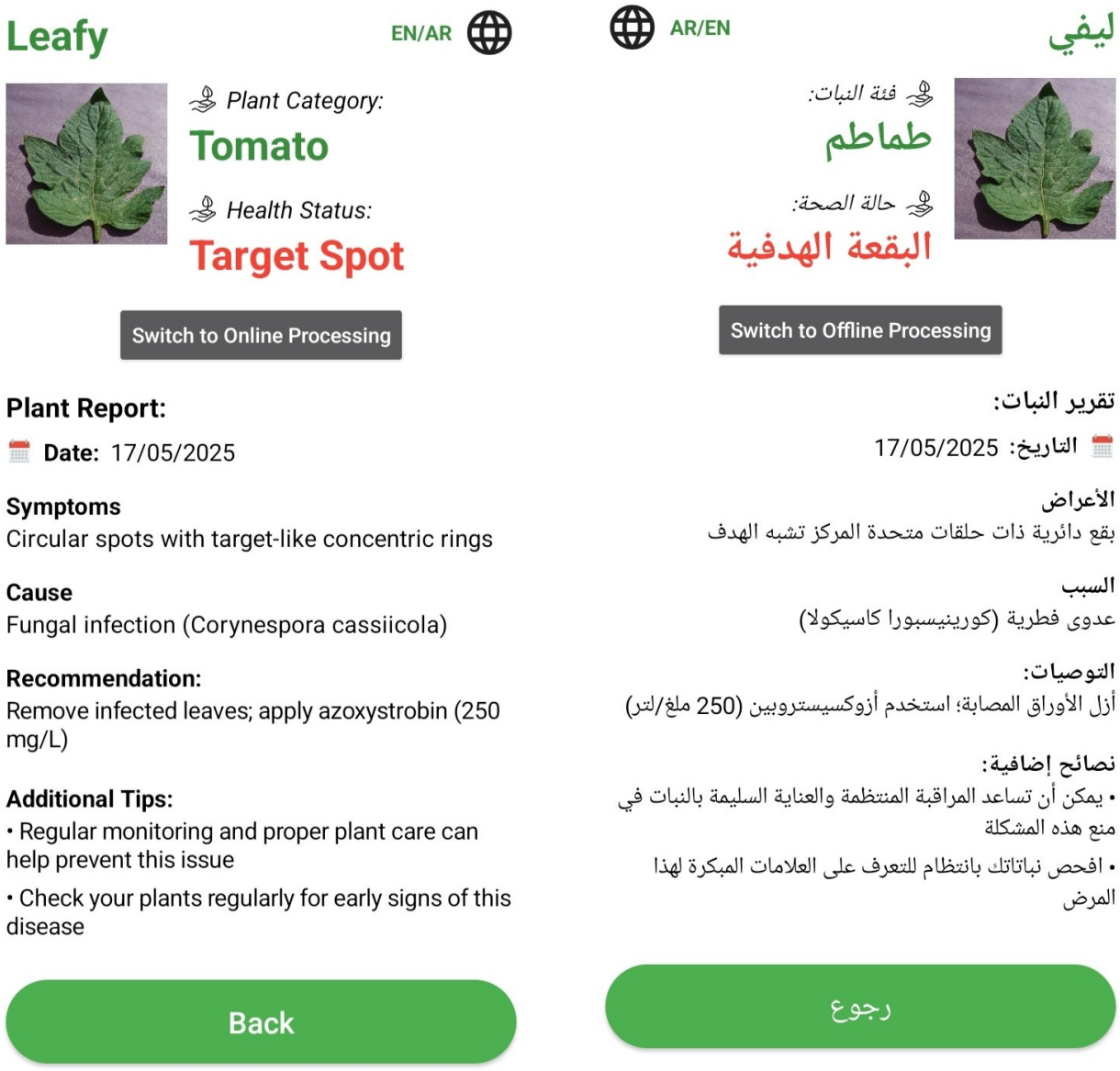


(a) English version showing location data and crop imagery.

(b) Arabic version maintaining consistent layout and functionality.

Figure 4.57: Process page for uploading or capturing plant images.

Following image analysis, Figure 4.58 demonstrates how diagnostic results are presented to users with actionable treatment recommendations. The report screens provide critical information about plant health status and specific remedies tailored to identified conditions. Figure 4.58(a) shows the English diagnostic report with detailed analysis and recommendations, while Figure 4.58(b) displays the Arabic diagnostic report maintaining consistent information structure.

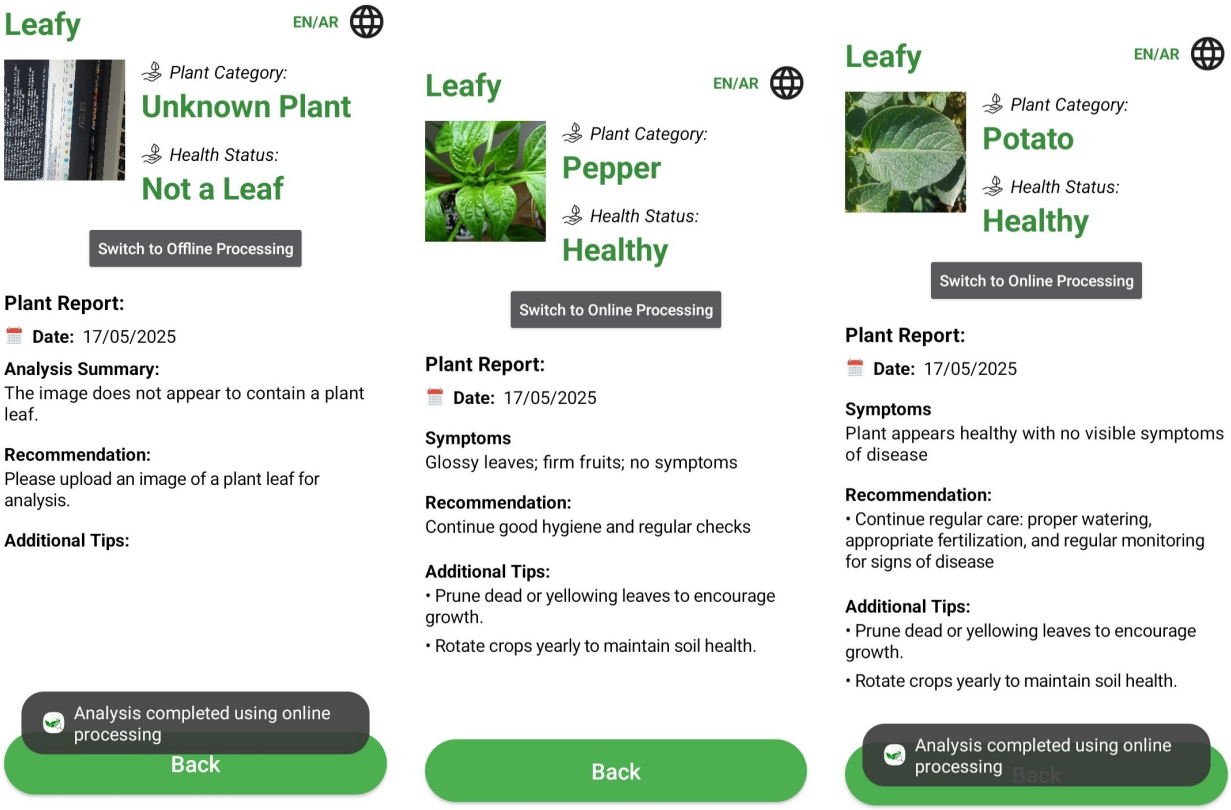


(a) English diagnostic report with detailed analysis and recommendations.

(b) Arabic diagnostic report maintaining consistent information structure.

Figure 4.58: Report screens showing disease diagnosis and treatment recommendations in both languages.

Extensive testing was conducted to ensure reliable performance across various input conditions including healthy plants, diseased plants, and invalid inputs, with results summarized in Figure 4.59. These tests, showcasing different user interface screens as seen in Figure 4.59a through Figure 4.59c, were designed to evaluate the application's performance across the full range of scenarios it might encounter in real-world use, ensuring robust performance in diverse field conditions.



(a) Welcome interface demonstration. (b) Image capture and upload functionality. (c) Diagnostic output presentation.

Figure 4.59: Testing results across different app workflows and scenarios.

As shown in Figure 4.59, the tests demonstrated the system's robustness in multiple scenarios:

- **Healthy Detection:** Correctly identified healthy plants with "No disease detected" messages, minimizing false positives that could lead to unnecessary treatments.
- **Disease Detection:** Accurately classified diseases such as Target Spot and Powdery Mildew with detailed treatment recommendations tailored to the specific condition.
- **Error Handling:** Effectively managed invalid or ambiguous images with appropriate error messages or fallback prompts, ensuring users understand when inputs are unsuitable for analysis.

4.5 Comparison Study

To evaluate the practical value of our system in the agricultural technology landscape, we conducted a comprehensive benchmarking against both commercial applications and recent academic research.

Our system was benchmarked against leading mobile agricultural application Plantix [118], AgriApp [119], and FarmRise [120] focusing on features that directly impact real-world usability and effectiveness. Table 4.10 summarizes this comparative analysis.

Feature	**Our System**	**Plantix**	**AgriApp**	**FarmRise**
Crop Coverage	Tomato, Potato, Pepper	30+ crops	20+ crops	10+ crops
OOD Detection	Yes (98.86%)	Basic filtering	No	No
Language Support	English, Arabic	10+ languages	English, Hindi	8+ languages
Treatment Recommendation	AI-generated contextual	Template-based	Static content	Template-based

Table 4.10: Feature comparison with existing agricultural applications.

This comparative analysis reveals several key advantages of our system. While our crop coverage is more focused than mature applications like Plantix, our system distinguishes itself through several primary innovations. Our Out-of-Distribution (OOD) detection system achieves 98.86% accuracy in distinguishing relevant plant images from irrelevant inputs, a sophisticated capability either absent or rudimentary in commercial alternatives. This significantly reduces false positives and improves reliability in uncontrolled field conditions. Additionally, our GPT-2 based recommendation engine generates adaptive, situation-specific guidance rather than the static or template-based advice offered by competitors, providing farmers with more personalized and effective treatment strategies. While our language support is currently limited to English and Arabic, it is strategically targeted to our primary user demographics, balancing comprehensiveness with usability.

To situate our results within the scientific literature, we compared our system against recent academic approaches in plant disease detection. Table 4.11 highlights the key features and improvements of our approach relative to existing research.

Our system demonstrates significant advancements over the current state of the art. Unlike many academic models that focus on single crops, our modular design achieves superior performance across multiple plant species through specialized architectures: DenseNet121 for

Dimension	Existing Approaches	Our System
Generalization and Robustness	Models such as [78, 85] perform well in controlled environments (99.35%) but decline dramatically in real-world conditions (31%).	Integrates Out-of-Distribution (OOD) detection with 98.86% accuracy to filter irrelevant inputs, maintaining consistent performance across varied field conditions.
Practical Deployment	Systems like [29, 80] demonstrate high accuracy but suffer performance degradation on resource-constrained devices. Kant et al. [79] achieved 92.25% accuracy with EfficientNetB3 but focused only on multilabel classification without end-to-end workflow.	Optimized MobileNetV2 and EfficientNet-B3 architecture enables efficient inference on standard mobile devices without performance compromise, while providing a complete pipeline from detection to recommendation.
Recommendation System	Most systems focus solely on classification without actionable guidance, creating a gap between detection and intervention.	Incorporates a fine-tuned GPT-2 based recommendation engine that transforms technical classifications into farmer-friendly, contextual treatment guidance with strong linguistic metrics (perplexity: 17.29, BLEU: 0.60, ROUGE-L: 0.78).
Multi-language Support	Academic implementations typically neglect accessibility factors critical for real-world adoption.	Bilingual interface.
Multilabel vs. Targeted Detection	Kant et al. [79] introduced multilabel classification to detect multiple diseases simultaneously but with moderate accuracy (92.25%) and without crop-specific optimization.	Employs specialized crop-specific models, improving accuracy to 98.8% for tomato diseases and 95.14% for potato diseases by targeting each crop's unique pathological characteristics.
Crop-Specific Accuracy	Prior studies report lower accuracy for specific crops (e.g., 91.2% for tomato in [85]).	Achieves improved accuracy 98.8% for tomato diseases and 95.14% for potato diseases through specialized architectures (DenseNet121 and ResNet) tailored to each crop's unique pathological characteristics.

Table 4.11: Comparison of findings against existing academic research.

tomato/potato (98.8% and 95.14% accuracy respectively, compared to 91.2% in [85]) and ResNet for pepper, each tailored to the unique pathological characteristics of their respective crop families.

While Kant et al. [79] pioneered multilabel classification with EfficientNetB3 (92.25% accuracy), our approach uses specialized crop-specific models that significantly improve accuracy while maintaining computational efficiency. Our system also addresses the dramatic field-setting accuracy decline observed by [78] through explicit OOD detection (98.86% accuracy), preventing non-plant image misclassification and resolving execution bottlenecks seen in implementations like [29].

By integrating CV-based detection with NLP-based recommendation generation, our system bridges the critical gap between technical identification and practical intervention a feature largely absent in existing academic work. This integration transforms disease classifications into actionable, context-aware treatment guidance in multiple languages. Our fine-tuned GPT-2 recommendation engine achieved impressive linguistic metrics (perplexity: 17.29, BLEU: 0.60, ROUGE-1: 0.82, ROUGE-2: 0.73, ROUGE-L: 0.78), indicating strong relevance and fluency in the generated treatment advice.

4.6 Conclusion

Our plant disease detection and recommendation system demonstrates significant advancements through its multi-model cascading architecture with specialized components for OOD detection (98.86% accuracy), crop classification (99.89% accuracy), and crop-specific disease classification (98.80% for tomato, 95.14% for potato). These results address fundamental limitations in current research, particularly regarding generalizability, deployment practicality, and end-user utility.

Our project revealed a clear relationship between dataset size and model performance, suggesting that agricultural AI development should prioritize collecting adequate samples per disease category before expanding crop coverage. By combining robust detection capabilities with contextually-aware recommendations and bilingual support, our system represents a significant step toward bridging the gap between academic research and practical agricultural solutions, making sophisticated AI capabilities accessible to farmers across varying technological and environmental contexts. The strong performance of our recommendation system demonstrates that AI-generated advice can match human expertise in specificity and relevance, a critical advancement for practical agricultural applications.

Chapter 5

Conclusion and Perspectives

This project successfully developed an intelligent, multi-model plant disease detection and recommendation system tailored for modern agriculture. The system achieved strong performance across multiple tasks, including image classification and language-based recommendation generation. For plant image classification, we reached up to 98.86% accuracy with MobileNetV2 for general detection, 99.89% with EfficientNet-B3 for crop identification, and 98.80% with DenseNet121 for tomato diseases. Potato diseases were detected with 95.14% accuracy, while pepper diseases reached 59.72% using ResNet50 due to limited training data. For natural language recommendations, a fine-tuned GPT-2 model achieved a perplexity of 17.29, a BLEU score of 0.60, ROUGE-1 of 0.82, ROUGE-2 of 0.73, and ROUGE-L of 0.78 indicating strong relevance and fluency in generated treatment advice.

Our solution combines image classification and natural language processing to filter irrelevant inputs, detect crop types, diagnose diseases, and deliver actionable treatment recommendations all within a mobile application. This design ensures advanced agricultural AI is accessible to farmers using standard smartphones.

The project resulted in several significant observations:

- Dataset size plays a critical role in model performance, with several hundred labeled samples per disease class necessary for robust results.
- The two-phase transfer learning strategy proved highly effective for agricultural image classification tasks.
- Fine-tuned language models like GPT-2 can generate context-aware, practical recommendations for farmers.

While we faced challenges such as limited data for pepper crop diseases, our results show that focused, specialized models trained with sufficient data can perform reliably in real-world agricultural settings.

The system is designed to generate significant positive outcomes for smallholder and commercial farmers alike:

- **Increased crop yields** through timely disease detection and intervention.
- **Reduced pesticide use** thanks to targeted and accurate diagnoses.
- **Democratization of agricultural expertise** via easy-to-use mobile interfaces.
- **Improved supply chain forecasting** by enabling early disease tracking and reporting.
- **Valuable field data** generation to support continuous model improvement and research.

In future work, we envision the system evolving along several promising directions:

- Enable disease progression tracking over time using temporal image analysis.
- Add multilingual support beyond English and Arabic to broaden accessibility.
- Expand classification capabilities to include a wider range of crops and pest-related conditions.
- Integrate treatment recommendations with local agricultural supply sources to streamline access to relevant products.
- Facilitate farmer-led knowledge exchange through in-app discussion forums and best-practice sharing.
- Explore integration with agricultural robotics for autonomous field monitoring and targeted spraying tasks 5.1.

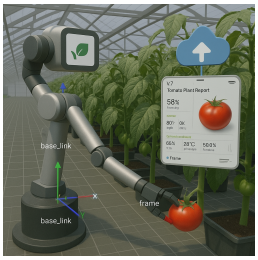


Figure 5.1: Conceptual imagination of a future agricultural robot.

Bibliography

[1] Food and Agriculture Organization of the United Nations. *Global Crop Losses due to Plant Diseases*. Accessed: 15 March 2025. 2024. URL: https://www.fao.org/home/fr.

[2] Direction des Statistiques Agricoles et des Systèmes d'Information (DSASI) République Algérienne Démocratique et Populaire Ministère de l'Agriculture, du Développement Rural et de la Pêche. *Synthèse des données agricoles – Wilaya de Biskra, 2023*. Internal document. Unpublished governmental report. Biskra, Algeria, 2023.

[3] Sjaak Wolfert et al. "Big data in smart farming – A review". In: *Agricultural Systems* 153 (2017), pp. 69–80. DOI: 10.1016/j.agsy.2017.01.023.

[4] Chandra Krintz et al. "SmartFarm: Improving Agriculture Sustainability Using Modern Information Technology". In: *KDD Workshop on Data Science for Food, Energy, and Water* (2016).

[5] Sravanthi Gunaganti and Nageswara Rao Moparthi. "An Efficient IoT Based Crop Disease Prediction and Crop Recommendation for Precision Agriculture". In: *Journal of Ambient Intelligence and Humanized Computing* (2024). Published online: 12 February 2024. DOI: 10.1007/s12652-024-04964-2.

[6] Food and Agriculture Organization of the United Nations. *The State of Food and Agriculture 2024*. Accessed: May 2025. 2024. URL: https://www.fao.org/publications/sofa/en/.

[7] H Charles J Godfray et al. "Food security: the challenge of feeding 9 billion people". In: *Science* 327.5967 (2010), pp. 812–818.

[8] Robert E. Evenson and Douglas Gollin. "Assessing the impact of the Green Revolution, 1960 to 2000". In: *Science* 300.5620 (2003), pp. 758–762. DOI: 10.1126/science.1087446.

[9] Andreas Kamilaris and Francesc X Prenafeta-Bold'u. "Deep learning in agriculture: A survey". In: *Computers and electronics in agriculture* 147 (2018), pp. 70–90.

[10] Robert Gebbers and Viacheslav I. Adamchuk. "Precision agriculture and food security". In: *Science* 327.5967 (2010), pp. 828–831. DOI: 10.1126/science.1183899.

[11] Christos Pylianidis, Sjoukje Osinga, and Ioannis N. Athanasiadis. "Introducing Digital Twins to Agriculture". In: *Computers and Electronics in Agriculture* 184 (2021), p. 105942. ISSN: 0168-1699. DOI: 10.1016/j.compag.2021.105942.

[12] VHive. *Smart Farming vs Traditional Farming*. Accessed: 2025-04-20. 2024. URL: https://vhive.buzz/smart-farming-vs-traditional-farming/.

[13] Amardeep Singh et al. "SMART Farming: Unleashing the Power of IoT and AI in Agriculture". In: *IEEE Access* 10 (2022), pp. 37487–37513.

[14] Achim Walter et al. "Smart farming is key to developing sustainable agriculture". In: *Proceedings of the National Academy of Sciences* 114.24 (2017), pp. 6148–6150.

[15] David B Lobell, Wolfram Schlenker, and Justin Costa-Roberts. "Climate trends and global crop production since 1980". In: *Science* 333.6042 (2011), pp. 616–620.

[16] Virginia Gorlinski. *The History of Agriculture*. Britannica Educational Publishing, 2013.

[17] Marcel Mazoyer and Laurence Roudart. *A history of world agriculture: from the neolithic age to the current crisis*. Monthly Review Press, 2006.

[18] Jonathan A. Foley et al. "Solutions for a cultivated planet". In: *Nature* 478 (2011), pp. 337–342. DOI: 10.1038/nature10452.

[19] Muhammad Ayaz et al. "Internet-of-Things (IoT)-Based Smart Agriculture: Toward Making the Fields Talk". In: *IEEE Access* 7 (2019), pp. 129551–129583. DOI: 10.1109/ACCESS.2019.2932609.

[20] Miguel A. Altieri. "Traditional Agriculture". In: *Encyclopedia of Biodiversity*. Ed. by Simon A. Levin. 2nd ed. Vol. 5. Academic Press, 2001, pp. 109–118. ISBN: 978-0-12-226865-6.

[21] Prabhu L. Pingali. "Green Revolution: Impacts, limits, and the path ahead". In: *Proceedings of the National Academy of Sciences* 109.31 (2012), pp. 12302–12308. DOI: 10.1073/pnas.0912953109.

[22] Eckhard C. Oerke. "Crop losses to pests". In: *Journal of Agricultural Science* 144.1 (2006), pp. 31–43. DOI: 10.1017/S0021859605005708.

[23] Jules Pretty. "Agricultural sustainability: concepts, principles and evidence". In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 363.1491 (2008), pp. 447–465. DOI: 10.1098/rstb.2007.2163.

[24] IT Munch Editorial Team. *How IT is Transforming Agriculture Practices*. Accessed: 2025-06-10. Image used from this article. IT Munch. n.d. URL: https://itmunch.com/it-in-transforming-agriculture-practices/.

[25] Pablo Tittonell. "Ecological intensification of agriculture sustainable by nature". In: *Current Opinion in Environmental Sustainability* 8 (2014), pp. 53–61. DOI: 10.1016/j.cosust.2014.08.006.

[26] Freepik Contributors. *Machine Learning in Agriculture - Free Images and Vectors*. Accessed: April 20, 2025. 2025. URL: https://www.freepik.com/free-photos-vectors/machine-learning-agriculture.

[27] Konstantinos G Liakos et al. "Machine learning in agriculture: A review". In: *Sensors* 18.8 (2018), p. 2674.

[28] Anna Chlingaryan, Salah Sukkarieh, and Brett Whelan. "Machine learning approaches for crop yield prediction and nitrogen status estimation in precision agriculture: A review". In: *Computers and electronics in agriculture* 151 (2018), pp. 61–69.

[29] Abdelhak Merizig et al. "Deep Learning-Based Tomato Disease Detection for Smart Agriculture". In: *IEEE Conference on Information and Communication Technology (ICT)*. Accessed: 15 March 2025. 2024. URL: https://doi.org/10.1007/s11042-023-17235-3.

[30] Ana Jimenez, David Posada, and Rodrigo Martinez. "Satellite image clustering in precision agriculture using supervised and unsupervised approaches". In: *International Journal of Engineering Technology* 7.4 (2018), pp. 108–112.

[31] Shaomin Zhang et al. "Plant disease identification based on deep learning algorithm in smart farming". In: *Sensors* 19.12 (2018), p. 2645.

[32] L. Zhang et al. "MMDGAN: a fusion data augmentation method for tomato-leaf disease identification". In: *Applied Soft Computing* 123 (2022), p. 108969.

[33] FAO. *The State of the World's Land and Water Resources for Food and Agriculture (SOLAW) – Managing Systems at Risk*. Food, Agriculture Organization of the United Nations, Rome, and Earthscan, London, 2011. URL: https://www.fao.org/4/i1688e/i1688e00.htm.

[34] Richard N Strange and Peter R Scott. "Plant disease: A threat to global food security". In: *Annual review of phytopathology* 43 (2005), pp. 83–116.

[35] Freepik Contributors. *Farming Challenges Images and Illustrations*. Accessed: April 20, 2025. 2025. URL: https://www.freepik.com/search?format=search&last_filter=query&last_value=farming+challenges&query=farming+challenges.

[36] Hrithik Paul et al. "A study and comparison of deep learning based potato leaf disease detection and classification techniques using explainable AI". In: *Springer Nature* (2023). Published online: 16 October 2023. DOI: 10.1007/s10586-023-04246-w.

[37] Nourhan Wael. "Potato Leaf Diseases Detection with 92% Accuracy". In: *Kaggle Notebooks* (2024). Accessed: 2024-06-15. URL: https://www.kaggle.com/code/nourhanwael7/potato-leaf-diseases-acc-92/notebook.

[38] Choyon Chandra Bonik et al. "A Convolutional Neural Network Based Potato Leaf Diseases Detection Using Sequential Model". In: *2023 International Conference for Advancement in Technology (ICONAT)*. Goa, India: IEEE, Apr. 2023. DOI: 10.1109/ICONAT57137.2023.10080063.

[39] Amanda Ramcharan et al. "Deep learning for plant disease detection and diagnosis". In: *Frontiers in Plant Science* 8 (2017), p. 1419.

[40] Jeffrey B. Jones et al. *Compendium of Tomato Diseases and Pests, Second Edition*. St. Paul, MN: APS Press, 2014. ISBN: 978-0-89054-424-2.

[41] Saman Fatima. "Tomato Leaf Disease Classification with 94% Accuracy Using Deep Learning". In: *Kaggle Notebooks* (2024). Accessed: 2024-06-15. URL: https://www.kaggle.com/code/samanfatima7/tomato-leaf-disease-94-accuracy/notebook.

[42] Alexander Uzhinskiy. *Pepper Disease Classification Dataset (DoctorP)*. Version 1.0. 2024. URL: https://www.kaggle.com/datasets/alexanderuzhinskiy/pepper-disease-classification-dataset-doctorp.

[43] Madhu Dhingra, Kalpana Jain, and Rishamjot Kaur. "Application of support vector machines for crop disease diagnosis: A review". In: *International Journal of Engineering Research Technology* 8.4 (2019), pp. 398–404.

[44] Ishtiak Hossain Sarker. "Machine Learning: Algorithms, Real-World Applications and Research Directions". In: *SN Computer Science* 2 (2021), p. 160. DOI: 10.1007/s42979-021-00592-x. URL: https://doi.org/10.1007/s42979-021-00592-x.

[45] Qiong Liu and Ying Wu. *Supervised Learning*. Jan. 2012. DOI: 10.1007/978-1-4419-1428-6_451.

[46] Wei-Meng Lee. "Supervised Learning Linear Regression". In: *Beginning Machine Learning in the Cloud with Python: Understand, Analyze, and Visualize Data*. Wiley, Apr. 2019, pp. 119–149. ISBN: 9781119545637. DOI: 10.1002/9781119557500.ch6.

[47] Ahlem Walha et al. "Deep Learning and Machine Learning Architectures for Dementia Detection from Speech in Women". In: *Expert Systems with Applications* 239 (2024), p. 122404. DOI: 10.1016/j.eswa.2024.122404.

[48] P. B. Padol and A. A. Yadav. "SVM classifier based grape leaf disease detection". In: *2016 Conference on Advances in Signal Processing (CASP)*. IEEE. 2016, pp. 175–179.

[49] Samreen Naeem et al. “An Unsupervised Machine Learning Algorithms: Comprehensive Review”. In: *IJCDS Journal* 13 (Apr. 2023), pp. 911–921. DOI: 10.12785/ijcds/130172.

[50] S. M. Javidan et al. “Diagnosis of grape leaf diseases using automatic K-means clustering and machine learning”. In: *Smart Agricultural Technology* 3 (2023), p. 100081.

[51] M. A. R. Nishad, M. A. Mitu, and N. Jahan. “Predicting and classifying potato leaf disease using K-means segmentation techniques and deep learning networks”. In: *Procedia Computer Science* 212 (2022), pp. 220–229. DOI: 10.1016/j.procs.2022.11.006.

[52] Padmanabha Y C A, Viswanath Pulabaigari, and Eswara B. “Semi-supervised learning: a brief review”. In: *International Journal of Engineering Technology* 7 (Feb. 2018), p. 81. DOI: 10.14419/ijet.v7i1.8.9977.

[53] Mohamed F.A. Hady and Friedhelm Schwenker. “Semi-supervised Learning”. In: *Handbook on Neural Information Processing*. Ed. by Monica Bianchini, Marco Maggini, and Lakhmi Jain. Vol. 49. Intelligent Systems Reference Library. Springer, 2013. Chap. 7. DOI: 10.1007/978-3-642-36657-4_7. URL: https://doi.org/10.1007/978-3-642-36657-4_7.

[54] Muddasar Naeem, Syed Rizvi, and Antonio Coronato. “A Gentle Introduction to Reinforcement Learning and its Application in Different Fields”. In: *IEEE Access* 8 (Jan. 2020), pp. 209320–209344. DOI: 10.1109/ACCESS.2020.3038605.

[55] Wang Qiang and Zhan Zhongli. “Reinforcement learning model, algorithms and its application”. In: *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*. 2011, pp. 1143–1146. DOI: 10.1109/MEC.2011.6025669.

[56] M. A. Moid and M. A. Chaurasia. “Transfer learning-based plant disease detection and diagnosis system using Xception”. In: *2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*. IEEE. 2021, pp. 1–5.

[57] M. Shaha and M. Pawar. “Transfer learning for image classification”. In: *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. IEEE. 2018, pp. 656–660. DOI: 10.1109/ICECA.2018.8474802.

[58] T. Talaei Khoei, H. Ould Slimane, and N. Kaabouch. “Deep learning: Systematic review, models, challenges, and research directions”. In: *Neural Computing and Applications* 35.31 (2023), pp. 23103–23124. DOI: https://doi.org/10.1007/s00521-023-08957-4.

[59] Karan Jagdale et al. “Artificial Intelligence and its Subsets: Machine Learning and its Algorithms, Deep Learning, and their Future Trends”. In: *JETIR* 9 (June 2022), pp. i112–i117. DOI: 10.6084/m9.jetir.JETIR2205914.

[60] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Cambridge, MA: MIT Press, 2016. ISBN: 978-0262035613. URL: https://www.deeplearningbook.org/.

[61] Samaya Madhavan and M. Tim Jones. "Deep learning architectures: The rise of artificial intelligence". In: *Computer Science Review* 38 (2020), p. 100289. DOI: 10.1016/j.cosrev.2020.100289.

[62] Unknown. "Recurrent Neural Network". In: *Artificial Intelligence for Renewable Energy Systems*. In subject area: Engineering. Elsevier, 2022.

[63] S. Thomas, A. Khanna, and K. Singla. "Indian Crop Yield Prediction using LSTM Deep Learning Networks". In: *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. Kharagpur, India: IEEE, 2022. DOI: 10.1109/ICCCNT54827.2022.9984407. URL: https://ieeexplore.ieee.org/document/9984407.

[64] Ishana Attri et al. "Crop Disease Detection using Deep Learning Techniques: A Review". In: *2022 International Conference on Intelligent Systems Design and Applications (ISDA)*. IEEE, 2022. DOI: 10.1109/ISDA2022.2022.9909888. URL: https://ieeexplore.ieee.org/document/9909888.

[65] R. V. Dharani, T. Kalaiselvi, and R. V. Dharani. "A Survey on Smart Agriculture: Crop Disease Detection Using Machine Learning". In: *2017 International Conference on Smart Technologies for Smart Nation (SmartTechCon)*. Bangalore, India: IEEE, 2017. DOI: 10.1109/SmartTechCon.2017.8358382. URL: https://ieeexplore.ieee.org/document/8020005.

[66] Dave Bergmann and Cole Stryker. *Qu'est-ce qu'un auto-encodeur ?* Consulté en mai 2025. IBM, n.d. URL: https://www.ibm.com/fr-fr/think/topics/autoencoder.

[67] Konstantinos P. Ferentinos. "Deep learning models for plant disease detection and diagnosis". In: *Computers and Electronics in Agriculture* 145 (2018), pp. 311–318. DOI: 10.1016/j.compag.2018.01.009.

[68] GeeksforGeeks. *ML - Classification vs Clustering*. Accessed: 2025-04-25. 2021. URL: https://www.geeksforgeeks.org/ml-classification-vs-clustering/.

[69] Jamin Rahman Jim et al. "Recent advancements and challenges of NLP-based sentiment analysis: A state-of-the-art review". In: *Natural Language Processing Journal* 6 (Mar. 2024), p. 100059.

[70] Alba Gutiérrez Domínguez et al. "Natural language processing of social network data for the evaluation of agricultural and rural policies". In: *Journal of Rural Studies* 109 (2024), p. 103341. DOI: 10.1016/j.jrurstud.2024.103341.

[71] Biao Zhao et al. "ChatAgri: Exploring potentials of ChatGPT on cross-linguistic agricultural text classification". In: *Neurocomputing* 557 (2023), p. 126708. DOI: 10.1016/j.neucom.2023.126708.

[72] Ananya B. Sai, Akash Kumar Mohankumar, and Mitesh M. Khapra. "A Survey of Evaluation Metrics Used for NLG Systems". In: *ACM Computing Surveys (CSUR)* 55.2 (2022), pp. 1–39. DOI: 10.1145/3485766.

[73] AX Semantics. *What is Natural Language Generation (NLG) & Where is it Used?* https://www.ax-semantics.com/en/blog/natural-language-generation-explained. Accessed: 2025-04-25. 2024.

[74] Vasso Marinoudi et al. "Large language models impact on agricultural workforce dynamics: Opportunity or risk?" In: *Smart Agricultural Technology* 9 (2024), p. 100677. DOI: 10.1016/j.atech.2024.100677.

[75] Vishal Sharma, Rohit Kumar, and Gaurav Singh Thakur. "Natural language generation in agricultural advisory systems: A comprehensive review". In: *Applied Artificial Intelligence* 36.1 (2022), p. 2114358.

[76] Ehud Reiter. "Machine Learning and Neural NLG". In: *Natural Language Generation*. Springer, Cham, 2024, pp. 49–70. DOI: 10.1007/978-3-031-68582-8_4.

[77] Tom B. Brown et al. "Language Models are Few-Shot Learners". In: *arXiv preprint arXiv:2005.14165* (2020). URL: https://arxiv.org/abs/2005.14165.

[78] Sharada P. Mohanty, David P. Hughes, and Marcel Salathé. "Using Deep Learning for Image-Based Plant Disease Detection". In: *Frontiers in Plant Science* 7 (2016), p. 1419. DOI: 10.3389/fpls.2016.01419.

[79] Vishnu Kant. "Enhancing Plant Disease Detection with EfficientNetB3: A Multilabel Classification Approach". In: *2024 Second International Conference on Intelligent Cyber Physical Systems and Internet of Things (ICoICI)*. Conference Date: 28-30 August 2024. Coimbatore, India: IEEE, Aug. 2024. DOI: 10.1109/ICoICI62503.2024.10696264.

[80] Kashif Shaheed, Imran Qureshi, Muhammad Z. Sajid, et al. "EfficientRMTNet An Efficient ResNet-50 and Vision Transformers Approach for Classifying Potato Plant Leaf Diseases". In: *Sensors* 23.23 (2023), p. 9516. DOI: 10.3390/s23239516. URL: https://www.mdpi.com/1424-8220/23/23/9516.

[81] Li Wang and Wei Liu. "An efficient deep learning model for tomato disease detection". In: *Plant Methods* 20.1 (2024), pp. 1–12. DOI: 10.1186/s13007-024-01188-1. URL: https://plantmethods.biomedcentral.com/articles/10.1186/s13007-024-01188-1.

[82] L. Aversano et al. “Tomato diseases classification based on VGG and transfer learning”. In: *2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*. IEEE. 2020, pp. 129–133. DOI: 10.1109/MetroAgriFor50201.2020.9277626.

[83] Manjunatha Shettigere Krishna et al. “Plant Leaf Disease Detection Using Deep Learning: A Multi-Dataset Approach”. In: *J* 8.1 (2025). Submission received: 8 November 2024 / Revised: 27 December 2024 / Accepted: 7 January 2025 / Published: 15 January 2025, p. 4. DOI: 10.3390/j8010004. URL: https://doi.org/10.3390/j8010004.

[84] J. Rashid et al. “Multi-level deep learning model for potato leaf disease recognition”. In: *Electronics* 10.17 (2021), p. 2064. DOI: 10.3390/electronics10172064.

[85] M. Agarwal et al. “ToLeD: Tomato leaf disease detection using convolution neural network”. In: *Procedia Computer Science* 167 (2020), pp. 293–301.

[86] Mahesh Thyluru Ramakrishna et al. “Leveraging EfficientNetB3 in a Deep Learning Framework for High-Accuracy MRI Tumor Classification”. In: *Computers, Materials & Continua* 81.1 (2024), pp. 867–883. DOI: 10.32604/cmc.2024.048462.

[87] Yaoshiang Ho and Samuel Wooke. “The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling”. In: *IEEE Access* 8 (2020), pp. 215910–215920. DOI: 10.1109/ACCESS.2020.3039406.

[88] Author. “A novel hybrid deep learning approach for plant disease detection using transfer learning and explainable AI”. In: *Information Sciences* 660 (2024), pp. 119–135. DOI: 10.1016/j.ins.2024.119135. URL: https://www.sciencedirect.com/science/article/pii/S0020025524007965.

[89] Davide Chicco and Giuseppe Jurman. “The Matthews Correlation Coefficient (MCC) is more informative than Cohen’s Kappa and Brier score in binary classification assessment”. In: *IEEE Access* 7 (2019), pp. 77963–77970. DOI: 10.1109/ACCESS.2019.2923355.

[90] Iuliana Dobre. “A Comparison Between BLEU and METEOR Metrics Used for Assessing Students within an Informatics Discipline Course”. In: *Procedia - Social and Behavioral Sciences* 180 (2015), pp. 305–312. DOI: 10.1016/j.sbspro.2015.02.121.

[91] Shaik Abdul Samad et al. “Advancing Abstractive Summarization: Evaluating GPT-2, BART, T5-Small, and Pegasus Models with Baseline in ROUGE and BLEU Metrics”. In: *Innovations in Cybersecurity and Data Science (ICICDS 2024)*. Lecture Notes in Networks and Systems. First Online: 13 December 2024. Springer, 2024, pp. 119–131. DOI: 10.1007/978-3-031-51452-8_9. URL: https://link.springer.com/chapter/10.1007/978-3-031-51452-8_9.

[92] S. Sharma and T. S. Rappaport. "A Survey of Machine Learning Techniques in Ad Hoc Networks". In: *IEEE Communications Surveys Tutorials* 11.3 (2009), pp. 92–102. DOI: 10.1109/COMST.2009.4907287. URL: https://ieeexplore.ieee.org/document/5380758.

[93] Kavita Ganesan. "ROUGE 2.0: Updated and Improved Measures for Evaluation of Summarization Tasks". In: *arXiv preprint arXiv:1803.01937* (Mar. 2018). URL: https://arxiv.org/abs/1803.01937.

[94] Sergey Vychegzhanin et al. "Controllable Story Generation Based on Perplexity Minimization". In: *Analysis of Images, Social Networks and Texts (AIST 2023)*. Vol. 14486. Lecture Notes in Computer Science. First Online: 12 March 2024. Springer, 2024, pp. 154–169. DOI: 10.1007/978-3-031-38827-3_12. URL: https://link.springer.com/chapter/10.1007/978-3-031-38827-3_12.

[95] Python Software Foundation. *Python Programming Language*. 2025. URL: https://www.python.org/ (visited on 06/09/2025).

[96] Oracle Corporation. *Java Programming Language*. 2025. URL: https://www.java.com/ (visited on 06/09/2025).

[97] *Figma: Collaborative interface design tool*. https://www.figma.com.

[98] Arnaud Roques. *PlantUML Language Reference Guide*. Version 1.2024.7. PlantUML. 2024. URL: https://plantuml.com/ (visited on 07/15/2024).

[99] Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, et al. "Array programming with NumPy". In: *Nature* 585.7825 (2020), pp. 357–362.

[100] pandas Development Team. *pandas: Python Data Analysis Library*. 2025. URL: https://pandas.pydata.org/ (visited on 06/09/2025).

[101] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[102] Gary Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[103] Albumentations Team. *Albumentations: Fast image augmentation library*. 2025. URL: https://albumentations.ai/ (visited on 06/09/2025).

[104] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95.

[105] Michael L Waskom. "Seaborn: statistical data visualization". In: *Journal of Open Source Software* 6.60 (2021), p. 3021.

[106] NLTK Project. *Natural Language Toolkit*. 2025. URL: https://www.nltk.org/ (visited on 06/09/2025).

[107] Explosion AI. *spaCy: Industrial-strength Natural Language Processing*. 2025. URL: https://spacy.io/ (visited on 06/09/2025).

[108] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. https://www.tensorflow.org/. 2015.

[109] Keras Team. *Keras: Deep Learning for humans*. 2025. URL: https://keras.io/ (visited on 06/09/2025).

[110] PyTorch Team. *PyTorch: An open source machine learning framework*. 2025. URL: https://pytorch.org/ (visited on 06/09/2025).

[111] *TensorFlow Lite*. https://www.tensorflow.org/lite.

[112] Ekaba Bisong. *Google Colaboratory*. Apress, Berkeley, CA, 2019, pp. 59–64.

[113] *Kaggle: Your Home for Data Science*. https://www.kaggle.com.

[114] *Android Studio*. https://developer.android.com/studio.

[115] *Google Cloud*. https://cloud.google.com/.

[116] Pallets Projects. *Flask: web development, one drop at a time*. 2025. URL: https://flask.palletsprojects.com/ (visited on 06/09/2025).

[117] *WeatherAPI*. https://www.weatherapi.com/.

[118] *Plantix*. https://plantix.net/en/. Mobile application for plant disease diagnosis and crop advisory. 2025.

[119] *AgriApp*. https://www.agriapp.com/. Mobile application providing agricultural advisory and market information. 2025.

[120] *FarmRise*. https://www.farmrise.in/. Digital platform supporting farmers with financial and crop advisory services. 2025.