



الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche
scientifique

جامعة سعد دحلب البلدية
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique



Master's Dissertation

Presented by

Lamia SARAOUI

&

Raouf ABDALLAH ELHIRTSI

To obtain a Master's degree in Telecommunication

Option: Networks and Telecommunications

Topic

Classification of Plant Disease Using Deep Learning

Proposed by: Dr. F. FODHIL

Co-Advisor: Dr. Hocine AIT SAADI

Dedication

I offer this humble piece of work to

My mother “Chahera”, you’re the best mother anyone could ask for in this life, your unconditional love, sacrifice, constant prayers, and delicate care was what kept me going.

My father “Ahmed”, the selfless man who gave me all the support in the world, and your confidence in my capabilities has laid the groundwork for my accomplishments.

I am profoundly thankful for the numerous ways in which both of you have influenced my path and encouraged me to strive for my objectives. I extend my appreciation to my sister; your affection has been a consistent source of solace throughout this expedition.

Furthermore, I extend my gratitude to my mentor, "Dr. Fodhil," for her priceless advice and vision.

my co-advisor, "Dr. Ait Saadi Hocine, » Your cheerful personality, work ethic, and sagacity were instrumental throughout my journey at the university, my deep gratitude.

I am thankful to all the esteemed professors in our department, whose commitment to education has motivated me. Your enthusiasm for knowledge has driven my pursuit of distinction.

I am grateful to all my relatives and friends, both near and far, who have supported me and offered invaluable assistance on this journey.

Raouf ABDALLAH ELHIRSI

Dedication

I dedicate this modest work to

My beloved family, whose unwavering support and love have been my greatest strength.

To my mother and father, for their endless encouragement, sacrifices, and unconditional love. Your belief in my potential has been the foundation of my success.

Mom, your nurturing spirit and tireless support have been my guiding light.

Dad, your strength has taught me the value of perseverance and hard work, I am deeply grateful for the countless ways you both have shaped my journey and inspired me to reach for my goals.

To my sisters; «Loubna», «Nesrine», and «Nada», for their constant motivation and companionship. Your support and love have been a source of comfort and encouragement throughout this journey.

To my grandparents.

To my close friends «Ayoub», «Merouane», «Zohir», «Chiraz», and «Maissa», for their support and understanding. Your friendship has been a pillar of strength during the challenging times. Your support has meant the world to me.

To my advisor, «Dr. Fodhil», and my co-advisor, «Dr. Ait Saadi Hocine», for their guidance and wisdom.

To all the good professors of our department, whose dedication to teaching has inspired me. Your passion for knowledge has fueled my pursuit of excellence.

To my dear cousins, «Lina», «Rania», and «Selma», for their love and encouragement.

To my uncle's wife, for her kindness and support.

To my second Family ITCommunity.

To all my relatives and friends who have been there for me, near and far, and who have provided invaluable assistance throughout this journey. Your belief in me has been a source of immense strength.

Thank you all for being a part of this incredible journey and for believing in me.

Lamia SARAOU

Acknowledgments

With immense gratitude, we extend our sincere thanks to God the Almighty for bestowing upon us the health and strength that empowered us to undertake and complete this modest work.

We extend our deepest gratitude to our advisor, Dr. Fodhil, for her invaluable guidance, wisdom, and patience throughout this research. Your contributions have been pivotal in shaping the direction and quality of our work.

A special thanks to our co-advisor, Dr. Ait Saadi Hocine, for his exceptional dedication, insightful advice, and continuous encouragement. Your support has been instrumental in our academic growth and success.

We also express our sincere thanks to the members of the jury for their time, effort, and valuable feedback, by agreeing to review our work.

To the CRTI team, particularly Abdelghani Zabel, your collaboration and assistance have been greatly appreciated. Your support has played a crucial role in the successful completion of this research.

To all the professors of our department. Your teaching, mentorship, and encouragement have been essential in our academic journey.

Finally, we express deep gratitude to all those who contributed to this endeavor, whether near or far. Your valuable insights, assistance, and collaboration played a pivotal role in shaping the outcome of this work. Each contribution, no matter how small, has left an indelible mark on our collective achievement.

ملخص: تشكل الأمراض النباتية تحدياً كبيراً للزراعة وتؤثر على صحة المحاصيل والأمن الغذائي العالمي. وتعد الطرق التقليدية للكشف عن الأمراض، التي تعتمد على الفحص اليدوي، عرضة للأخطاء وعدم الكفاءة. تستكشف هذه الأطروحة تطبيق التعلم العميق (DL) في الكشف عن الأمراض النباتية، مع التركيز على ثلاثة نماذج: MobileNetV2، وAgriNetBoost، وVision Transformer (ViT). وقد اختيرت هذه النماذج لكفاءتها في البيئات المحدودة الموارد مثل الأجهزة المحمولة. تقوم الدراسة بتقييم دقة النماذج وسرعتها وكفاءتها الحسابية في هذا المجال وتنفيذ حل عملي باستخدام إطار عمل Streamlit للنشر في الوقت الحقيقي. تتمثل الأهداف في تعزيز قدرات الكشف عن الأمراض ودعم الممارسات الزراعية المستدامة.

الكلمات المفتاحية: أمراض النبات، الزراعة، الأمن الغذائي العالمي، التعلم الآلي (ML)، التعلم العميق (DL)، MobileNetV2، LightGBM، محول الرؤية (ViT).

Résumé: Les maladies des plantes représentent un défi important pour l'agriculture, car elles affectent la santé des cultures et la sécurité alimentaire mondiale. Les méthodes traditionnelles de détection des maladies, qui reposent sur l'inspection manuelle, sont sujettes aux erreurs et à l'inefficacité. Cette thèse explore l'application de l'apprentissage profond (DL) dans la détection des maladies des plantes, en se concentrant sur trois modèles: MobileNetV2, LightGBM et Vision Transformer (ViT). Ces modèles sont choisis pour leur efficacité dans des environnements à ressources limitées, tels que les appareils mobiles. L'étude évalue la précision, la vitesse et l'efficacité de calcul des modèles dans ce domaine et met en œuvre une solution pratique utilisant le cadre Streamlit pour un déploiement en temps réel. Les objectifs sont d'améliorer les capacités de détection des maladies et de soutenir les pratiques agricoles durables.

Mots clés: Maladies des plantes, agriculture, sécurité alimentaire mondiale, apprentissage automatique (ML), apprentissage profond (DL), MobileNetV2, LightGBM, Vision Transformer (ViT).

Abstract: Plant diseases pose a significant challenge to agriculture, affecting crop health and global food security. Traditional disease detection methods, reliant on manual inspection, are prone to errors and inefficiencies. This thesis explores the application of Deep Learning (DL) in detecting plant diseases, focusing on three models: MobileNetV2, AgriNetBoost, and the Vision Transformer (ViT). These models are chosen for their efficiency in resource-constrained environments such as mobile devices. The study evaluates the models' accuracy, speed, and computational efficiency in this field and implements a practical solution using the Streamlit framework for real-time deployment. The objectives are to enhance disease detection capabilities and support sustainable agricultural practices.

Keywords: Plant diseases, agriculture, global food security, Machine Learning (ML), Deep Learning (DL), MobileNetV2, AgriNetBoost, Vision Transformer (ViT).

Lists of acronyms and abbreviations

AI	Artificial Intelligence
ANN	Artificial neural networks
API	Application Programming Interface
CPU	Central Processing Unit
CRISPR	Clustered Regularly Interspaced Short Palindromic Repeats
DL	Deep Learning
DMTK	Distributed Machine Learning Toolkit
FAO	Food and Agriculture Organization
GAP	Global Average Pooling
GBT	Gradient Boosting Tree
GELU	Gaussian Error Linear Unit
GIS	Geographic Information Systems
GPS	Global Positioning Systems
GPU	Graphic Processing Unit
GUI	Graphical User Interface
IDE	Integrated Development Environment
IDM	Integrated Disease Management
IFPRI	International Food Policy Research Institute
JAX	Just Another XLA (Accelerated Linear Algebra)
LightGBM	Light Gradient Boosting Machine
LLMs	Large Language Models
LSTMs	Long Short-Term Memories
ML	Machine Learning

MLP	Multilayer Perceptron
MSE	Mean Squared Error
MT	Machine Translation
NLP	Natural Language Processing
NMT	Neural Machine Translation
ONNX	Open Neural Network Exchange
PIL	Python Imaging Library
RAM	Random Access Memory
ReLU	Rectified Linear Unit
RGB	Red, Green, Blue (color model)
RNA	Ribonucleic acid
RNAi	Ribonucleic acid interference
RTX	Ray Tracing Texel eXtreme (NVIDIA's graphics cards)
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
ToMV	Tomato mosaic virus
UAVs	Unmanned Aerial Vehicles
UNDP	United Nations Development Programme
USD	United state dollar
ViT	Vision Transformer
XGBoost	eXtreme Gradient Boosting
1D	One dimension
2D	Two dimensions

Table of contents

General introduction	1
Chapter I: State of The Art – Plant Diseases	4
I.1. Introduction.....	4
I.2. Plant diseases	4
I.2.1. Types of plant disease	5
I.3. Economic and environmental impact of plant diseases.....	11
I.4. Plant Disease Management	12
I.4.1. Principle of control.....	12
I.4.2. Integrated Disease Management (IDM).....	13
I.4.3. Recent Advances in Plant Disease Management	14
I.5. Precision agriculture.....	15
I.5.1. Definition	15
I.5.2. Diverse technologies in precision agriculture	16
I.6. Conclusion	19
Chapter II: State of the Art - Machine Learning & Deep Learning	21
II.1. Introduction.....	21
II.2. Machine Learning	21
II.2.1. Machine Learning approaches	22
II.3. Gradient Boosting Tree.....	23
II.3.1. LightGBM.....	23
II.4. Lightweight Models.....	24
II.5. Neural Network.....	25
II.5.1. Biological Neural Network.....	25
II.5.2. Artificial Neural Network.....	26
II.6. Deep Learning.....	27
II.6.1. Convolutional Neural Networks (CNNs).....	28
II.6.2. Transfer Learning	33
II.6.3. Transformers.....	35
II.7. Lightweight Models in Plant Disease Detection.....	37
II.8. Conclusion	39
Chapter III: Fine-tuning and Evaluation Metrics	41
III.1. Introduction	41
III.2. Hyperparameters.....	41
III.2.1. CNN hyperparameters	41
III.2.2. Transformers hyperparameters	43
III.2.3. LightGBM hyperparameters.....	44

III.3. Evaluation metrics	44
III.3.1. Confusion matrix	45
III.3.2. Accuracy	45
III.3.3. Precision	45
III.3.4. Recall	46
III.3.5. F1 score	46
III.4. Conclusion	46
Chapter IV: Design, Implementation & Results	48
IV.1. Introduction	48
IV.2. Working environment	48
IV.3. Presentation of the languages used.....	48
IV.3.1. Introducing the Python language.....	49
IV.3.2. Introducing the imported libraries	49
IV.4. Methodology	51
IV.4.1. System architecture	51
IV.4.2. Used dataset	51
IV.4.3. Dataset preparation.....	54
IV.4.4. Used models	58
IV.5. Results and Discussion.....	64
IV.5.1. MobileNetV2 Results.....	64
IV.5.2. Google vit Results	68
IV.5.3. AgriNetBoost	72
IV.6. Practical implementation.....	75
IV.6.1. Model Selection and Training	75
IV.6.2. Web application development.....	75
IV.6.3. Test and validation	78
IV.7. Results and Comparison.....	81
IV.8. Conclusion.....	83
General Conclusion	84
Annex.....	85
Bibliography	90

Figures list

Chapter I: State of The Art – Plant Diseases

<i>Figure I.1.</i> Schematic representation of plant diseases and pathogens.....	5
<i>Figure I.2.</i> Potato leaf infected with early blight.....	6
<i>Figure I.3.</i> Pepper leaf infected with bacterial leaf spot.....	7
<i>Figure I.4.</i> Tomato leaf infected with mosaic virus.....	8
<i>Figure I.5.</i> Potato plant with severe nematode damage.....	8
<i>Figure I.6.</i> Life cycle of a root parasitic plant, <i>Orobanche minor</i>	9
<i>Figure I.7.</i> Environmental stressors affecting plants	10
<i>Figure I.8.</i> Comprehensive impacts of plant diseases on global agriculture, economy, and food security	12
<i>Figure I.9.</i> Precision agriculture: Integrating technology for optimized farming	16
<i>Figure I.10.</i> Satellite scanning a field	17
<i>Figure I.11.</i> Smartphone usage in precision agriculture.....	19

Chapter II: State of the Art - Machine Learning & Deep Learning

<i>Figure II.1.</i> Supervised learning	22
<i>Figure II.2.</i> Level-wise growth vs leaf-wise growth	24
<i>Figure II.3.</i> Knowledge distillation	25
<i>Figure II.4.</i> Biological Neural Network	26
<i>Figure II.5.</i> Artificial Neural Network	27
<i>Figure II.6.</i> Convolutional Neural Networks architecture.....	28
<i>Figure II.7.</i> Convolution operation.....	29
<i>Figure II.8.</i> Three types of pooling operations.....	20
<i>Figure II.9.</i> ReLU activation function operation process	29
<i>Figure II.10.</i> Timeline of CNN models	32
<i>Figure II.11.</i> The MobileNetV2 network architecture	33
<i>Figure II.12.</i> Traditional machine learning vs Transfer learning	34

Chapter IV: Design, Implementation & Results

<i>Figure IV.1.</i> Detailed Workflow Diagram.....	51
<i>Figure IV.2.</i> Visual dataset for identification and classification of plant diseases.....	52
<i>Figure IV.3.</i> Visualization of data augmentation.....	57
<i>Figure IV.4.</i> MobileNetV2 model visualization	58
<i>Figure IV.5.</i> Google ViT Model visualization.....	62
<i>Figure IV.6.</i> AgriNetBoost model visualization.....	64
<i>Figure IV.7.</i> MobileNetV2 Training and Validation Performance.....	65
<i>Figure IV.8.</i> MobileNetV2 Model evaluation on the test dataset.....	65
<i>Figure IV.9.</i> MobileNetV2 plot training and validation accuracy and loss values.....	65
<i>Figure IV.10.</i> MobileNetV2 Confusion Matrix.....	66
<i>Figure IV.11.</i> MobileNetV2 Classification Report.....	67
<i>Figure IV.12.</i> Training and Validation Performance - Google ViT	68

Figure IV.13. Google ViT Model Evaluation on the test dataset	68
Figure IV.14. Google ViT Plot training and validation accuracy and loss values	69
Figure IV.15. Google ViT Confusion Matrix	70
Figure III.16. Google ViT Classification Report	71
Figure III.17. Training and Validation Performance - AgriNetBoost.....	72
Figure III.18. AgriNetBoost Confusion Matrix	73
Figure III.19. AgriNetBoost Classification Report	74
Figure III.20. User Interface Screenshots	76
Figure III.21. Choosing a model	76
Figure III.22. The diagnostic.....	77
Figure III.23. Test images	78
Figure III.24. Classification and confidence results in a healthy Potato leaf.....	79
Figure III.25. Classification and confidence results on a Pepper bell bacterial spot infected leaf	80
Figure III.26. Classification and confidence results on a Tomato late blight leaf	80
Figure III.27. Classification and confidence results on a tomato leaf infected with YLCV	81

Table list

Chapter I: State of The Art – Plant Diseases

Table I.1. A comparison between satellites and drones in agriculture..... 18

Chapter II: State of the Art - Machine Learning & Deep Learning

Table II.1. Lightweight models in plant disease detection 39

Chapter III: Fine-tuning and Evaluation Metrics

Table III.1. Confusion matrix 45

Chapter IV: Design, Implementation & Results

Table IV.1. Distribution of PlantVillage dataset samples 53

Table IV.2. Distribution of PlantVillage dataset samples 55

Table IV.3. MobileNetV2 model summary 59

Table IV.4. Google ViT Model Summary..... 61

Table IV.5. Comparative table of results of all models..... 81

General introduction

Plant diseases pose a significant threat to global agricultural productivity, potentially leading to up to 40% annual crop losses worldwide, according to the (FAO) [1]. Early detection of these diseases is crucial, as it can reduce yield losses by as much as 50%, as indicated by research in the Journal of Plant Pathology [2]. The impact is especially severe in developing countries, where over 1 billion people rely on agriculture for their livelihoods, contributing to increased food insecurity, as highlighted by the (IFPRI) [3]. Studies from the (UNDP) underscore that these countries bear the brunt of plant disease impacts, with up to 80% of their rural population dependent on agriculture [4]. Therefore, there is a growing emphasis on identifying and managing plant diseases to safeguard agricultural productivity and mitigate their detrimental effects on global food security.

Traditionally, disease detection relied heavily on manual inspection, a method prone to human error and inefficiencies. However, recent advancements in technology, particularly in the fields of machine learning (ML) and deep learning (DL) enabling automated and accurate identification of plant diseases based on visual symptoms and patterns.

This thesis explores the application of ML and DL techniques in the domain of plant disease detection. Specifically, it investigates the efficacy of lightweight models designed to operate efficiently on resource-constrained environments, including mobile devices and remote agricultural settings. By evaluating and comparing various ML and DL models, this research aims to identify the most effective approach for enhancing disease detection capabilities.

This thesis investigates the application of state-of-the-art ML and DL techniques in the detection of plant diseases, with a specific focus on three distinct models: MobileNetV2, a DL model optimized for mobile platforms; LightGBM, a gradient boosting framework known for its speed and accuracy; and the Vision Transformer (ViT), a DL model that has shown promising results in image classification tasks by leveraging self-attention mechanisms. These models are evaluated for their ability to analyze visual symptoms of plant diseases captured through image data. Furthermore, the research explores the practical implementation of these models using the Streamlit framework a versatile tool for building and deploying data-centric web applications.

Streamlit facilitates the development of interactive and intuitive interfaces, enabling seamless access and utilization of disease detection models by stakeholders in agriculture. The deployment phase focuses on optimizing model performance, and user interface design ensuring practical applicability in agricultural settings.

The study's objectives are twofold: first, to assess and compare the performance of these models in terms of accuracy, speed, and computational efficiency for plant disease detection; second, to develop a practical solution using the Streamlit framework for real-time deployment in agricultural settings. By achieving these objectives, this research contributes to advancing disease monitoring capabilities and supporting sustainable agricultural practices.

The first chapter provides a comprehensive overview of plant diseases, their types, causes, and impacts, along with management strategies and the role of precision agriculture.

The second chapter will delve into the evolution and applications of ML and DL, focusing on core principles like supervised and unsupervised learning. It will highlight neural networks such as CNNs and Transformers. Additionally, it will explore the role of lightweight models in enabling efficient AI applications on resource-constrained devices.

The third chapter explores fine-tuning deep learning models for plant disease detection, focusing on essential hyperparameters for CNNs, Transformers, and LightGBM, and evaluation metrics including confusion matrices, accuracy, precision, recall, and F1 score to assess classification model performance with imbalanced data.

The fourth chapter will feature the case study that will showcase our work. We will build three lightweight models, discuss their outcomes, implement them in a web application, and conclude with a comparative analysis of these models.



Chapter I

State of The Art

Plant Diseases

Chapter I: State of The Art – Plant Diseases

I.1. Introduction

Plant diseases represent a significant issue within the realm of agriculture, carrying extensive economic and environmental implications. This section delves into the intricate facets of plant diseases, thoroughly examining their origins, manifestations, and the wide-ranging effects they have on agriculture and ecosystems, along with strategies for their control. Furthermore, we will explore both conventional and innovative methods for handling plant diseases, emphasizing efficient techniques for mitigating their impact.

Upon completing this section, readers will possess a robust comprehension of plant diseases and their mitigation, equipped to further explore these pivotal areas of research. Concluding this section, a cutting-edge analysis will be presented on the utilization of advanced technologies in detecting plant diseases, merging classical plant pathology with contemporary approaches like precision agriculture and machine learning.

I.2. Plant diseases

A plant disease can be defined as any condition that disrupts the normal growth and development of a plant, leading to a decrease in its economic or aesthetic value. This disruption affects the functioning of the plant, resulting in lower yields or reduced quality, which in turn impacts the income of farmers, reduces food supplies, and may increase consumer prices. Unlike injuries caused by immediate factors like insect feeding or mechanical damage, plant diseases develop gradually over time. Symptoms of the disease typically become apparent only after several days of its onset [5].

These diseases can be caused by either living agents, such as microorganisms and parasitic plants (referred to as pathogens or biotic agents), or non-living agents, such as environmental factors, inadequate nutrition, and chemical substances (referred to as abiotic agents). Some diseases involve a combination of both pathogens and abiotic factors.

Figure I.1 displays a diagram showing the complex interactions between pathogens, environmental conditions, and host plant characteristics in plant diseases. It highlights the importance of considering these three factors in plant pathology. The visual representation helps understand how pathogens, environmental factors, and host vulnerabilities contribute to and affect plant diseases.

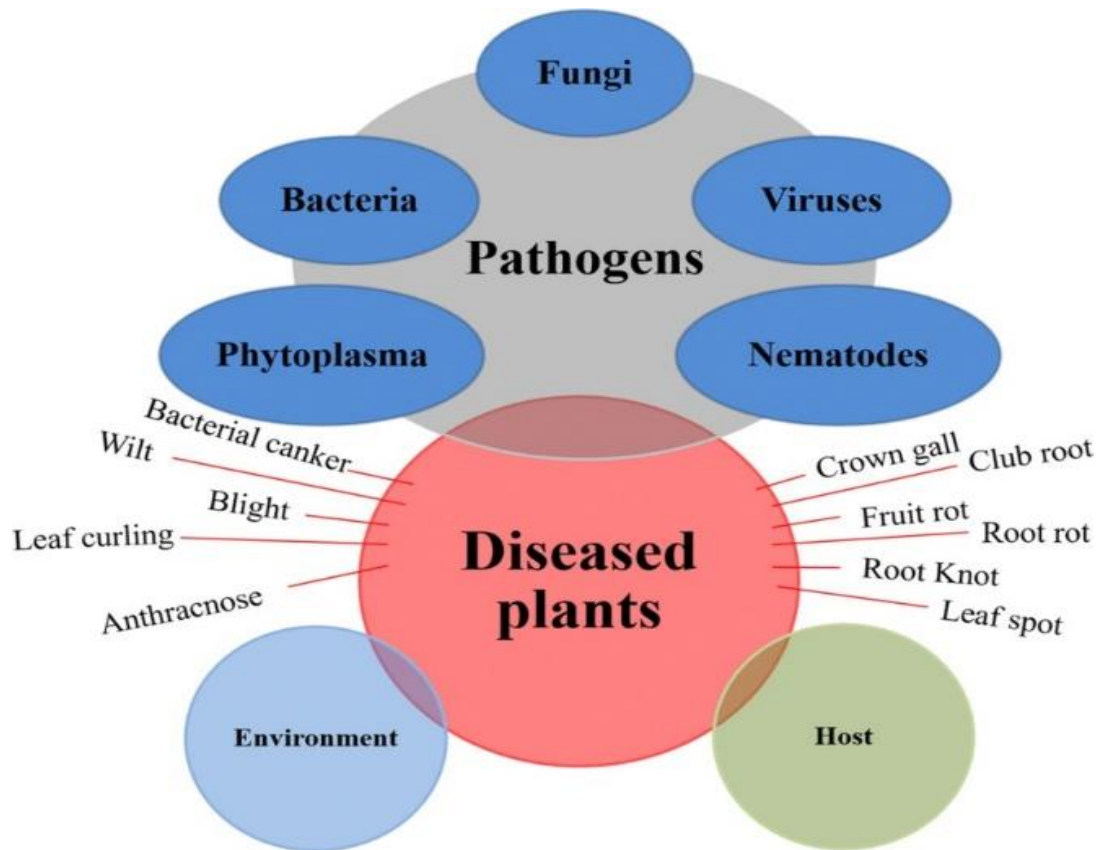


Figure I.1. Schematic representation of plant diseases and pathogens [6]

I.2.1. Types of plant disease

Manifestations of plant diseases often appear as discernible marks or lesions on leaves, stems, flowers, or fruits. Each disease or pest exhibits unique characteristics aiding in diagnosis, changes in leaf color, texture, edges, and size serve as telltale signs of declining plant health [7].

Plant diseases can be broadly categorized based on the primary cause of infection, whether it is infectious or non-infectious in nature [8].

I.2.1.1. Infectious plant disease

Infectious plant diseases result from living agents called pathogens, which can spread from infected plants to healthy ones. These pathogens include microorganisms like nematodes, fungi, bacteria, and mycoplasmas, as well as viruses and viroids, which rely on living cells for reproduction [5].

a. Fungi

Fungi are composed of hyphae, tiny filaments made of small cells only visible under a microscope. The tangled mass of hyphae is known as mycelium. Fungi can form colonies on improperly stored bread or vegetables. Some fungi grow inside plants or organic debris, while others become mushrooms. Fungi produce spores that spread through wind, water, or people to germinate on plants. Hyphae can enter plants through openings or wounds, damaging tissues with toxins or enzymes. Fungicides are used to kill fungi causing plant diseases [5]. In figure I.2 shows a potato leaf infected with early blight, caused by *Alternaria solani fungus*, affects tomato and potato plants. Symptoms include small brown spots with rings on lower leaves. Disease spreads outwards leading to yellowing, withering, and death of leaves. Infection can also impact stem, fruit, and upper part of the plant inducing severe damage to crops [9].



Figure I.2. Potato leaf infected with early blight [10]

b. Bacteria

Bacteria, as single-celled microorganisms, possess rigid cell walls and reproduce through binary fission, with certain strains exhibiting rapid division rates under optimal nutrient conditions. This accelerated multiplication, coupled with the secretion of toxins and enzymes, contributes significantly to the degradation of plant tissues. Plant diseases stemming from bacterial infection are predominantly attributed to rod-shaped strains, which enter plants through wounds or natural openings. Additionally, some bacterial species produce growth-regulating substances, leading to aberrant tissue growth. Management strategies often involve the use of bactericides, and specialized antibiotics targeting bacterial pathogens, to mitigate bacterial-induced plant diseases [5].

Chapter I: State of The Art - Plant Diseases

Figure I.3 represents a Bacterial leaf spot, attributed to “*Xanthomonas campestris pv. vesicatoria*”, that poses a significant threat to pepper cultivation. This rod-shaped bacterium has a remarkable ability to survive in seeds and plant residue across seasons. Its diverse strains exhibit specificity towards certain pepper cultivars, manifesting disease symptoms in those varieties. The impact can be profound, with early defoliation and fruit malformation leading to substantial crop losses. As the disease progresses, its control becomes increasingly challenging, often resulting in irreversible damage to the plants [11].



Figure I.3. Pepper leaf infected with bacterial leaf spot [11]

c. Viruses and Viroid

Tiny invaders called viruses and viroids wreak havoc on plants. These microscopic entities, much smaller than 300 nanometers, hijack plant cells using their RNA genetic material. Viruses, with shapes like rods or spheres, manipulate the cell's machinery to make more viruses, while viroids are even simpler, just RNA molecules. Both disrupt plant growth, causing diseases. Spread through insects, contaminated tools, or even by us, these plant pathogens are a challenge to control, as effective treatments are still being developed. [5]. Figure I.4 shows tomato leaf infected with mosaic virus, it is a single-stranded RNA virus from the family *Virgaviridae*, genus *Tobamovirus*, infecting plants globally. It has a broad host range, affecting vegetables, flowers, and seedlings. ToMV leads to mosaic disease in various crop plants, posing a significant threat to worldwide tomato production [12].



Figure I.4. Tomato leaf infected with mosaic virus [13]

d. Nematodes

Nematodes, small eel-shaped worms often invisible to the naked eye, use a tiny style to extract nutrients from plants. They reproduce by laying eggs that hatch into larvae, which go through four molts to become adults. Some nematodes can complete their life cycle in under 30 days. Nematodes that affect plant roots are particularly important, though many species can feed on different parts of the plant. Substances used to eradicate nematodes are known as nematicides [5]. Figure I.5 presents a visual representation of a potato plant suffering from a severe nematode infection, evident through the wilting, discoloration, stunted growth, and implication of root knotting.



Figure I.5. Potato plant with severe nematode damage

e. Parasitic Plants

Parasitic plants are angiosperms that propagate through authentic seeds. The majority of these parasitic plants possess altered root-like structures that adhere to plant tissues for the acquisition of nutrients and water, yet lack root systems capable of absorbing nutrients from the soil. By utilizing nutrients that would typically be assimilated by the host plant, parasitic plants diminish the vigor of the host. Mistletoes represent one of the most commonly observed parasitic plant species. In certain instances, in which mistletoes afflict forest trees, the plant tissues may become so disarrayed that the integrity of the wood is compromised, resulting in the deformation of branches [5]. Figure I.6 illustrates the life cycle of a root parasitic plant, *Orobanche minor*, (a) Seed germination is elicited by host-derived stimulants, including strigolactones. (b) Seedling attaches to host root with haustoria. (c–d) Parasite tubercles grow underground for several weeks or months before emergence of the flowering shoots. (e) The parasite produces a large number of seeds, which remain viable for many years in soil [14].

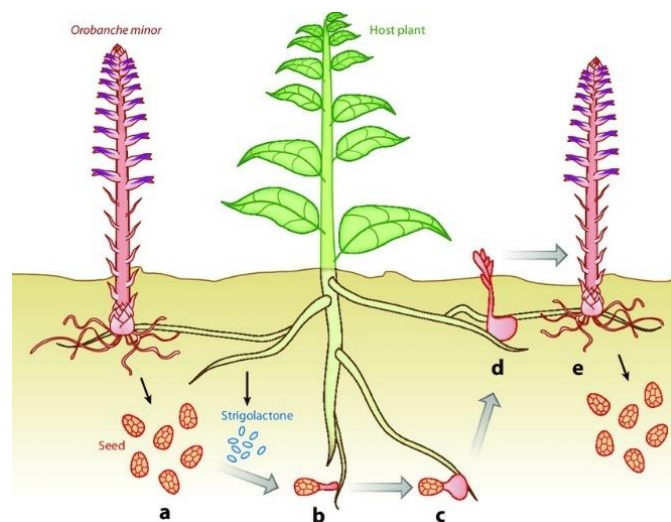


Figure I.6. Life cycle of a root parasitic plant, *Orobanche minor* [14]

f. Disease Complexes

Plant diseases, particularly those caused by soilborne pathogens, frequently involve multiple pathogens. Co-occurrence of pathogens can lead to increased plant damage or death. Plants may resist one parasite but struggle when attacked by multiple. Plant-disease interaction occurs when the combined damage of multiple pathogens exceeds the damage caused by individual ones. For instance, tomatoes with root-knot may succumb faster to the Granville-wilt bacterium. Peach trees have short lives due to cold injury, bacterial canker, improper pruning, and root-rotting organisms [5].

I.2.1.2. Non-infectious Plant Diseases

Non-infectious plant diseases are caused by non-living agents, typically environmental, nutritional, or chemical factors. Environmental extremes in temperature, moisture, or light can hinder plant development, while nutritional imbalances can lead to disease-like symptoms. Diseases induced by chemicals often result from inappropriate soil pH, fertilizer and pesticide misuse, or air contamination. Air pollutants from vehicular and industrial sources have increasingly been linked to plant health deterioration, with acid rain further impacting plant species and water quality in lakes [5]. Figure I.7 effectively conveys how abiotic factors—sunlight, water, temperature, soil composition, and physical weather events—interact to influence plant health. It underscores the importance of managing environmental conditions to mitigate non-infectious plant diseases, which are critical for sustaining healthy plant growth and agricultural productivity.

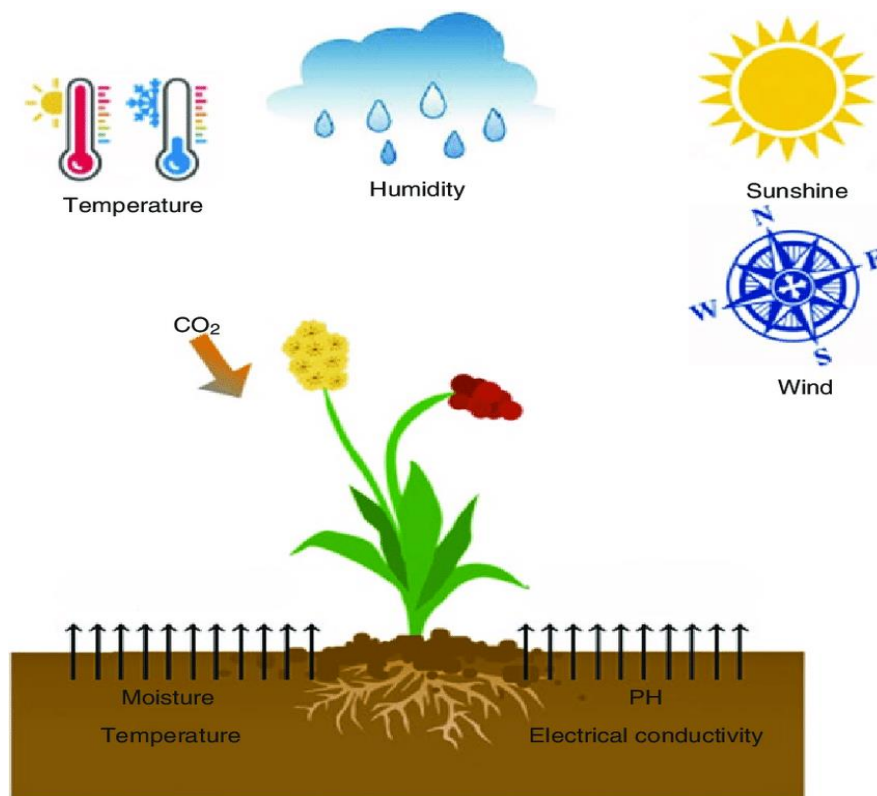


Figure I.7. Environmental stressors affecting plants [15]

I.3. Economic and environmental impact of plant diseases

Plants play a crucial role in the sustenance of the human species, accounting for 80% of the food sources we rely on and producing 98% of the oxygen crucial for our survival. Recent years have witnessed a substantial increase in the global trade of agricultural goods, tripling in size to a value of USD 1.7 trillion, particularly driven by the growth in emerging economies and developing countries. Nonetheless, the challenge of meeting the future demand for food presents a formidable obstacle, necessitating a 60% rise in agricultural output by 2050 to cater to a larger and wealthier population. This endeavor is further complicated by the pervasiveness of plant diseases, which annually lead to losses of up to 40% in global food crops, translating to trade deficits exceeding USD 220 billion. Furthermore, the effects of climate change worsen these challenges by jeopardizing crop yields and nutritional value, with increasing temperatures fostering the spread of plant diseases to new regions and earlier time frames [1]. Additionally, research by the International Food Policy Research Institute (IFPRI) highlights that agricultural productivity losses due to plant diseases contribute to increased food insecurity, affecting over 1 billion people worldwide [3]. Moreover, a study published in the Journal of Plant Pathology found that early detection of plant diseases can lead to a 50% reduction in yield losses [2]. Furthermore, studies conducted by the United Nations Development Programme (UNDP) reveal that developing countries, heavily reliant on agriculture, bear the brunt of plant disease impacts, with up to 80% of the rural population depending on agriculture for their livelihoods [4]. Figure I.8 presented below comprehensively illustrates the multifaceted impacts of plant diseases across various critical categories. Each category highlights a different dimension of the broader socio-economic and environmental ramifications of plant diseases.

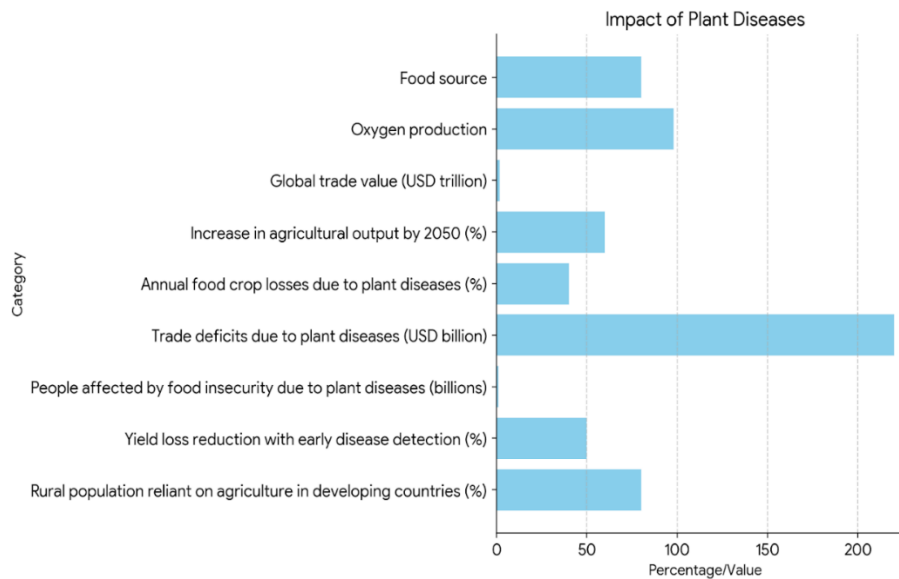


Figure I.8. Comprehensive impacts of plant diseases on global agriculture, economy, and food security

I.4. Plant Disease Management

Once the cause of a disease is accurately identified, effective strategies for its management and control can be developed. Over the past century, extensive research has thoroughly explored pathogens, diseases, and various management techniques. This wealth of knowledge now provides a robust foundation for enhancing disease control initiatives. Efficiently managing plant diseases is not only a scientific necessity but also an economic imperative, crucial for preventing devastating outbreaks and catastrophic famines [16].

I.4.1. Principle of control

Control of a plant disease involves reducing the extent of damage incurred. Attaining absolute control is uncommon, yet achieving profitable control, where the increased yield outweighs the expenses of chemicals and labor, is feasible. The foundational principles of control encompass exclusion, eradication, protection, resistance, and therapy.

a. Exclusion pertains to the prevention of pathogens from infiltrating and establishing in uninfected areas like gardens, states, or countries. Home gardeners practice exclusion by utilizing certified seeds or plants, inspecting bulbs before planting, discarding questionable ones, potentially treating seeds or tubers, and avoiding visibly diseased specimens from nurseries or dealers. At the state and country level, exclusion involves imposing quarantines or legal restrictions.

Chapter I: State of The Art - Plant Diseases

b. Eradication refers to eradicating a pathogen once it has taken hold on a plant or within a garden. It involves removing diseased specimens or parts, such as rouging for virus diseases or pruning cankered tree limbs, as well as practices like cultivation, deep ploughing, crop rotation, and disinfection using chemicals or heat treatment.

c. Protection entails placing a protective barrier between the susceptible part of the host plant and the pathogen. This often involves applying protective sprays or dust pre-emptively, eliminating inoculating agents like insects, or installing physical barriers like windbreaks.

d. Resistance is managed through the development of resistant cultivars. Resistant cultivars have been present since ancient times. The process of natural selection has historically eliminated unsuitable varieties, however, since approximately 1890, humans have accelerated this process through intentional breeding, selection, and propagation of plants resistant to prevalent diseases.

e. Therapy involves controlling the pathogen by inoculating or treating the plant with substances that deactivate it. Chemotherapy entails using chemicals to deactivate the pathogen, whereas thermotherapy or heat is occasionally employed to deactivate or impede virus development in infected plant tissues, enabling the growth of new tissue free from the pathogen [17].

I.4.2. Integrated Disease Management (IDM)

Integrated Disease Management (IDM) in agriculture is a multifaceted approach that combines various control strategies to minimize the impact of pathogens on crop and livestock production. This approach, which includes biological, cultural, physical, and chemical control methods, is effective and sustainable in both fish farming [18] and plant disease management [19], [20], [21]. However, successful implementation of IDM in developing countries requires a supportive policy environment and the adoption of participatory approaches [20]. Furthermore, the choice of specific disease management strategies should consider their impact on soil and crop health, as well as on the broader agricultural and non-agricultural environments [19]. IDM can target specific diseases like potato late blight, with goals of reducing inoculum and enhancing host resistance. Successful disease management relies on research about plant pests, crop resistance, and environmental conditions. Continuous research is needed for effective control measures [5].

Chapter I: State of The Art - Plant Diseases

Integration of Diverse Tactics: IDM utilizes various disease management tactics, such as:

- a. Cultural practices:** Techniques like crop rotation, sanitation, and proper irrigation to create a less hospitable environment for pathogens.
- b. Resistant crop varieties:** Selecting plant varieties with natural resistance to specific diseases.
- c. Biological control:** Introducing beneficial organisms like predators or parasites to control pathogen populations.
- d. Chemical control:** Using pesticides judiciously and only when necessary, considering economic thresholds and environmental impact.
- e. Monitoring systems:** Regularly monitoring crops or populations for disease presence and severity.
- f. Synergistic Effect:** By combining these tactics, IDM aims to create a synergistic effect, where the combined impact is greater than the sum of individual strategies.

Decision-Based Approach: IDM emphasizes monitoring and scouting to determine the need for intervention. Treatments are only applied, when necessary, based on economic thresholds and the specific disease situation.

I.4.3. Recent Advances in Plant Disease Management

Some of the recent advance techniques and approaches used in Plant Disease Management:

Integrated Management Strategies: These approaches combine various tools and practices to enhance disease control and minimize environmental impact [22].

Marker-Assisted Molecular Breeding and Biotechnological Approaches: Marker-assisted breeding allows precise selection of disease-resistant traits, while biotechnological methods like CRISPR-Cas and RNAi offer targeted gene modifications [23]. These tools aid in detecting and identifying plant pathogens, crucial for effective disease management [24].

Nanotechnology Applications: Nanotechnology is increasingly used in plant disease management, with nano-based sensors detecting pesticide residues and mycoflora [16], Nanoparticles can enhance plant defense mechanisms [25].

Electronic Monitoring (E-monitoring): This technology provides real-time data on disease spread, enabling timely interventions [26]. Furthermore, disease forecasting models and computer simulations are crucial for surveillance and mapping [16].

In summary, a multifaceted approach that integrates modern technologies, sustainable practices, and eco-friendly alternatives is essential for effective plant disease management.

I.5. Precision agriculture

Agricultural and forestry experts traditionally identify diseases and pests, but this method is subjective, time-consuming, and inefficient. Modernizing agriculture to increase efficiency and profitability has led to significant changes in practices. These include intensified land use, new crop management techniques, modern cultivars, changes in food preferences and policies, evolving trade regulations, and increased international movement of goods and people. These changes have diverse impacts on the agricultural system [27].

I.5.1. Definition

Precision farming, also known as precision agriculture, is a comprehensive system that optimizes agricultural production through the application of crop information, advanced technology, and management practices [28]. It involves the use of information and technology-based farm management systems to identify, analyze, and manage spatial and temporal variability within fields for optimum productivity and profitability [29]. This approach is based on variable soil and microclimate conditions within fields and is guided by technologies such as Global Positioning Systems (GPS), digital maps, Geographic Information Systems (GIS), and computers on-board agricultural vehicles [30]. Recent advancements in precision farming include the integration of technical equipment, automation, and robotics to achieve autonomous robotized systems [31]. Thus, customizing farming practices to accommodate this variability can enhance outcomes and mitigate undesirable effects [32]. Figure I.9 depicts various technological advancements in precision agriculture, highlighting the integration of modern technologies such as smartphones, drones, sensors, and automated machinery. These tools facilitate real-time data collection and analysis, enhancing decision-making processes and optimizing agricultural practices.

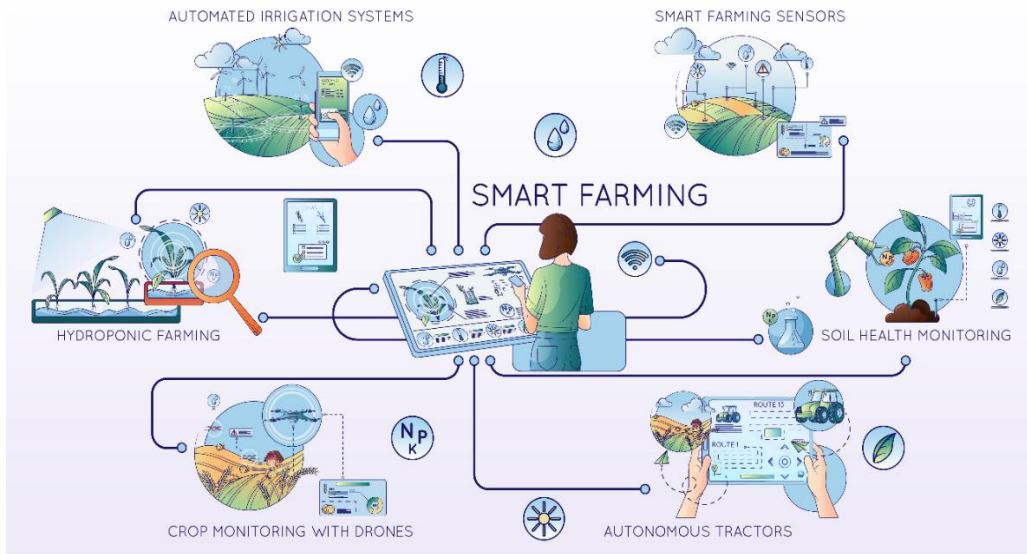


Figure I.9. Precision agriculture: Integrating technology for optimized farming [33]

I.5.2. Diverse technologies in precision agriculture

Precision agriculture leverages a wide range of cutting-edge technologies to improve the efficiency and sustainability of farming practices. These technologies include:

a. Satellite and remote sensing technologies

Agriculture depends on factors such as soil composition, weather, temperature, rainfall, crop growth stages, and topography. Satellites and space-borne technologies allow for easy monitoring of these variables from computer displays, providing data for strategic agricultural interventions. The use of satellites in agriculture is growing, evolving from data collection to precise farming activities like using GPS-equipped tractors for harvesting. Satellites primarily generate accurate geospatial data on farmlands and crops using multiple satellites and trilateration. Equipped with efficient sensors, satellites monitor and measure key agricultural variables, making them essential tools in modern farming practices [34]. Figure I.10 shows a satellite using remote sensing to gather data from an agricultural field.

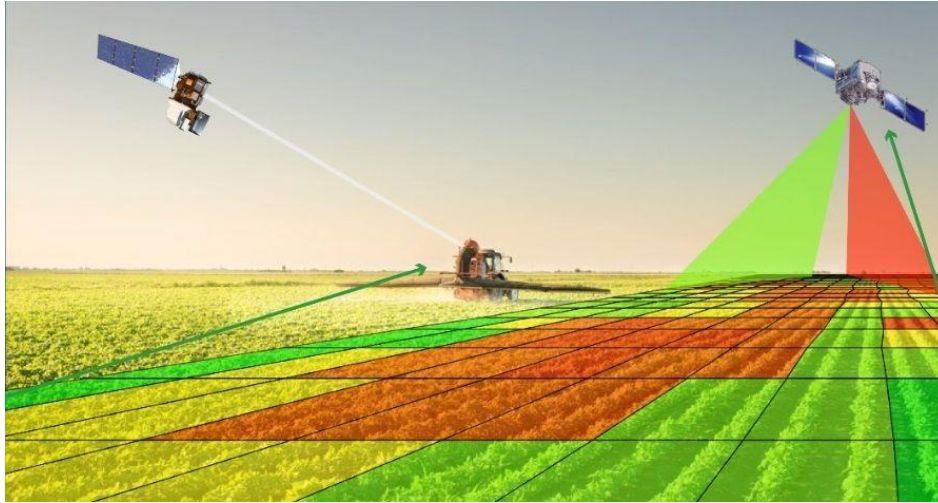


Figure I.10. Satellite scanning a field [35]

b. Drones and UAVs

Drones are increasingly utilized within the realm of agriculture as a component of a comprehensive sustainable farming strategy, enabling agronomists and farmers to facilitate the optimization of processes while acquiring crucial insights on their crops through detailed data analysis and topographical assessments. Specifically, the monitoring of crops is streamlined by the data obtained from agricultural drones, which is subsequently utilized in the development and execution of ongoing improvements such as adjustments in fertilizer application or drainage positioning. By utilizing GPS coordinates at different intervals throughout the journey, as opposed to labor-intensive and time-consuming data collection methods, food can be accurately traced from the farm to the consumer's plate. Within agriculture, drones can be integrated with various imaging technologies including hyperspectral, multispectral, and thermal imaging, among others, to provide farmers with timely and location-specific details regarding crop vitality, fungal outbreaks, growth impediments, and more. Unmanned Aerial Vehicles (UAVs) prove to be highly efficient in meticulously monitoring vast stretches of agricultural terrain, considering factors such as slope and elevation, for example, in order to determine the most optimal planting schedule. Notably, the high-resolution data obtained from drones can be leveraged to assess the fertility levels of crops, enabling agronomists to precisely apply fertilizers, minimize wastage, and develop or enhance irrigation systems [36].

Chapter I: State of The Art - Plant Diseases

Table I.1 presents a comparative analysis of drones and satellites in the context of agricultural precision, highlighting several key differences, while satellites offer extensive coverage and are suitable for large-scale monitoring, drones provide higher spatial resolution, up-to-date data, and detailed 3D modeling capabilities. This makes drones particularly valuable for precise, localized agricultural applications.

Category	Satellite	Drone
Cost	High, per use	Low, cost of the drone
Speed	Wait for satellite	Deploy on command
Temporal Resolution	Out-of-date	Up-to-date
Spatial Resolution	25 cm resolution	Centimeter-level accuracy with RTK
Map Area	Unlimited	3 km ² in one flight
3D Models and Point Clouds	No	Yes

Table I.1. A comparison between satellites and drones in agriculture [37]

c. Smartphone application

The outbreak of Artificial Intelligence (AI) has significantly advanced precision agriculture, revolutionizing how farmers monitor and manage their crops. Leveraging the power of AI, smartphone applications have emerged as indispensable tools in modern farming practices. Smartphone applications play a crucial role in precision agriculture by providing farmers with advanced tools for monitoring and managing their crops. These applications leverage the various sensors embedded in smartphones, such as motion, image, environment, and position sensors, to support real-time farming activities efficiently and at a low cost. The applications can be categorized into several groups, including agriculture management information, resource information, calculators, news, weather updates, and m-government services (mobile government). Specific apps focus on tasks like crop monitoring, disease detection, pest management, and soil analysis, offering functionalities that range from basic data recording to sophisticated data analytics and machine learning capabilities.

Chapter I: State of The Art - Plant Diseases

This integration of smartphone technology into agriculture not only enhances the accuracy and efficiency of farming practices but also provides a cost-effective solution for small-scale and resource-constrained farmers. Figure I.11 illustrates the application of smart farming technology, showcasing a smartphone displaying agricultural analytics. The phone screen presents data on crop yield, soil erosion, moisture stress, and soil cultivability. Surrounding icons represent key aspects of agriculture such as location tracking, weather, water management, plant health, pest control, and crop growth.



Figure I.11. Smartphone usage in precision agriculture [38]

I.6. Conclusion

This chapter explores plant diseases, their types, causes, and significant economic and environmental impacts. It outlines various management strategies and introduces precision agriculture, highlighting the role of advanced technologies in disease monitoring and management. Integrating these modern technologies with traditional methods enhances agricultural practices, ensuring crop health and contributing to increased productivity and sustainability.

The following chapter will discuss Machine learning, Deep Learning concepts, and some lightweight models used in plant disease detection.

Chapter II

State of the Art - Machine
Learning & Deep Learning

Chapter II: State of the Art - Machine Learning & Deep Learning

II.1. Introduction

Machine Learning (ML) and Deep Learning (DL) are two of the most transformative technologies of our time, and they are playing increasingly important roles in our lives. They are used in a variety of areas such as healthcare, business, finance, education, and more, and have the potential to drastically alter how we interact with the world around us.

This chapter will provide an overview of these two fields, including their history, applications, and recent advancements. We will also discuss the differences between machine learning and deep learning, and how they are used in various industries. By the end of this chapter, readers should have a solid understanding of machine learning and deep learning and be ready to dive deeper into these exciting areas of study.

Toward the end of this chapter, we will present a state-of-the-art review on the application of Machine Learning and Deep Learning in plant disease detection.

II.2. Machine Learning

In 1959, Arthur Samuel, a prominent computer scientist and pioneer in the field of machine learning, defined machine learning as the “field of study that gives computers the ability to learn without being explicitly programmed” [39]. This definition emphasizes the autonomous nature of learning in machines.

Tom Mitchell, in his book dedicated to machine learning, characterized machine learning as "the study of computer algorithms that allows computer programs to automatically improve through experience" [40]. Learning, according to Mitchell, occurs when a computer program demonstrates improved performance in a defined set of tasks by learning from a specific set of experiences.

Machine learning, a subset of artificial intelligence, equips computers and machines with the capability to assimilate information from data sets and utilize this knowledge to execute similar tasks without the need for explicit programming.

The categorization of ML algorithms commonly encompasses supervised and unsupervised learning, along with additional variations like reinforcement learning and semi-supervised learning [41].

II.2.1. Machine Learning approaches

II.2.1.1. Supervised learning

Supervised learning is the learning process where the output variable is known. The training process requires the explicit use of the variable's output. In supervised learning, data has labels. To put it simply, you are aware of the outcome you are aiming for. The algorithm has a specific goal or result that it aims to forecast from a set of factors that influence it (independent variables). By utilizing these specified parameters, a function is created that links inputs to expected results. Figure II.1 depicts the training process, which continues until the model reaches the desired level of accuracy on the training data. In the current era, data is essential for training and improving your models' learning capabilities. Nevertheless, the proportion of data changes depending on its size [42].

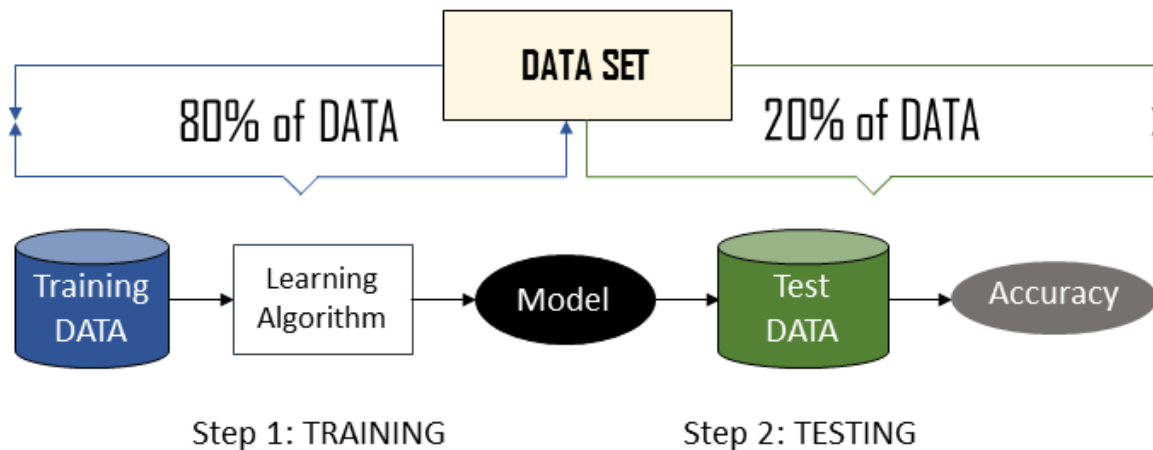


Figure II.1. Supervised learning [42]

II.2.1.2. Unsupervised learning

Unsupervised learning is a type of machine learning that identifies patterns and structures in data without any prior knowledge of the outcome. It clusters data based on similarities and identifies hierarchies among those clusters. It does not rely on pre-labeled datasets, making it useful for tasks with unknown outcome variables. It is commonly used for customer segmentation, identifying clusters among groups of customers [42].

II.2.1.3. Semi-supervised learning

In semi-supervised learning, a small amount of labeled data is combined with a large amount of unlabeled data during training. Some supervision guides the model despite the majority of data being unlabeled, leading to improved learning accuracy [41].

Chapter II: State of The Art - Machine Learning & Deep Learning

II.2.1.4. Self-supervised learning

Self-supervised learning, dubbed “the dark matter of intelligence” [43] is a variant of unsupervised learning used to address image annotation challenges. It involves creating pretext tasks to replace human-labeled data with computed pseudo-labels from raw input. In this process, the model learns one input part from another part. The main concept is to generate supervisory signals by interpreting unlabeled data in an unsupervised manner during the initial iteration [44].

II.2.1.5. Reinforced learning

Reinforcement learning involves agents learning from interactions. It includes agents, environments, and rewards. Agents observe, act, and sequentially receive feedback. The aim is to learn an optimal policy for maximizing rewards [44].

II.3. Gradient Boosting Tree

GBT is a tree-based ensemble algorithm. Boosting is used to create a strong learner from weak learners in GBTs. Decision trees are trained sequentially with each succeeding tree reducing the error of the previous one. Residuals of the previous model are used to fit the next model in GBTs. The residual correction process is repeated for a set number of iterations determined by cross-validation until residuals are minimized [45].

II.3.1. LightGBM

On October 17, 2016 [45], LightGBM emerged as a formidable competitor in the realm of tree-based gradient boosting within Microsoft’s Distributed Machine Learning Toolkit (DMTK) initiative. Its design prioritizes speed and distribution, leading to accelerated training speed and minimal memory consumption. The platform encompasses support for GPU utilization, parallel learning, and adept handling of substantial datasets.

Numerous benchmarks and experiments with public datasets have demonstrated LightGBM's superior speed and accuracy over XGBoost. Through the implementation of histograms for binning continuous features, LightGBM gains various performance benefits, such as decreased memory usage, lower computation costs for split gain calculation, and reduced communication overhead in parallel learning. Furthermore, LightGBM enhances its performance by employing histogram subtraction on a node's sibling and parent to compute the node's histogram.

Chapter II: State of The Art - Machine Learning & Deep Learning

Decision tree training typically follows two primary strategies: level-wise and leaf-wise growth. While level-wise growth represents the conventional approach in most tree-based ensembles like XGBoost, LightGBM has introduced the leaf-wise growth strategy. In contrast to level-wise growth, leaf-wise growth often achieves faster convergence and lower loss rates [45]. The difference between Level-wise growth and leaf-wise growth is presented in Figure II.2.

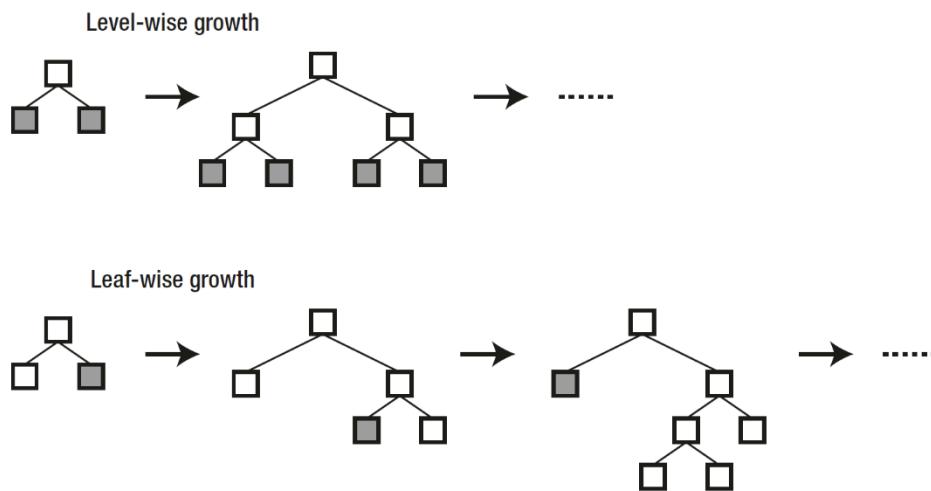


Figure II.2. Level-wise growth vs leaf-wise growth [45]

II.4. Lightweight Models

Lightweight machine learning models are created to be efficient and resource-friendly, making them ideal for deployment on devices with limited resources. They are crucial for deploying machine learning solutions on resource-constrained devices like mobile phones, edge devices, and embedded systems. These models balance complexity and computational efficiency to achieve reasonable performance while minimizing the computational burden. Here are some lightweight models we will discuss in detail in this chapter; MobileNet, CNNLite, TinyML, MobileViT. Key characteristics include:

- **Reduced Parameters:** These models have fewer learnable parameters than their larger counterparts, reducing memory footprint and inference time.
- **Simplified Architectures:** Streamlined architectural choices, such as depth-wise separable convolutions, group convolutions, or factorized convolutions, enable efficient feature extraction.

Chapter II: State of The Art - Machine Learning & Deep Learning

- **Quantization and Pruning:** Techniques like quantization (reducing the precision of weights and activations) and pruning (removing unimportant connections) further reduce model size.
- **Knowledge Distillation:** Lightweight models can be trained using knowledge distillation, where a larger pre-trained model (teacher) guides the training of a smaller model (student), see Figure II.3 [46].

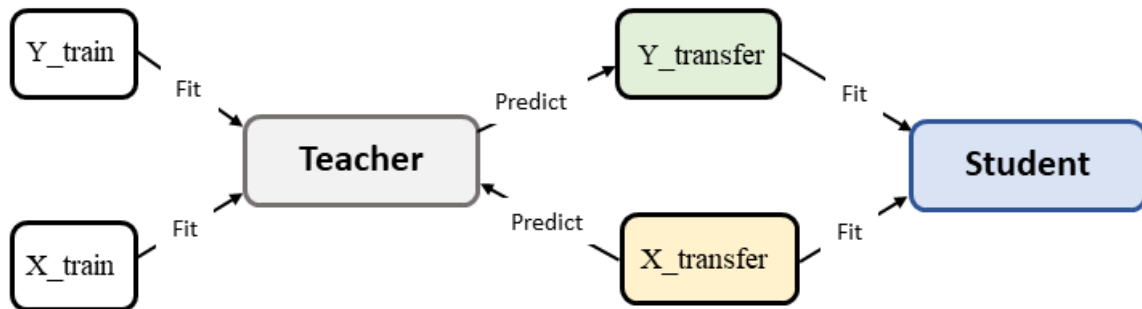


Figure II.3. Knowledge distillation [46]

II.5. Neural Network

II.5.1. Biological Neural Network

The brain is composed of interconnected nerve cells called neurons. It has around 10 billion neurons and 60 trillion synapse connections. Multiple neurons working together make the brain faster than computers. Neurons are simple but powerful, with a soma, dendrites, and an axon as illustrated in Figure II.4. Neurons communicate through electrochemical reactions and exhibit plasticity. Neural networks learn and process information globally. Strengthened connections lead to learning, while weakened ones diminish. Neural networks learn through experience, like biological neural networks. Emulating biological neural networks is attempted in computer systems [47].

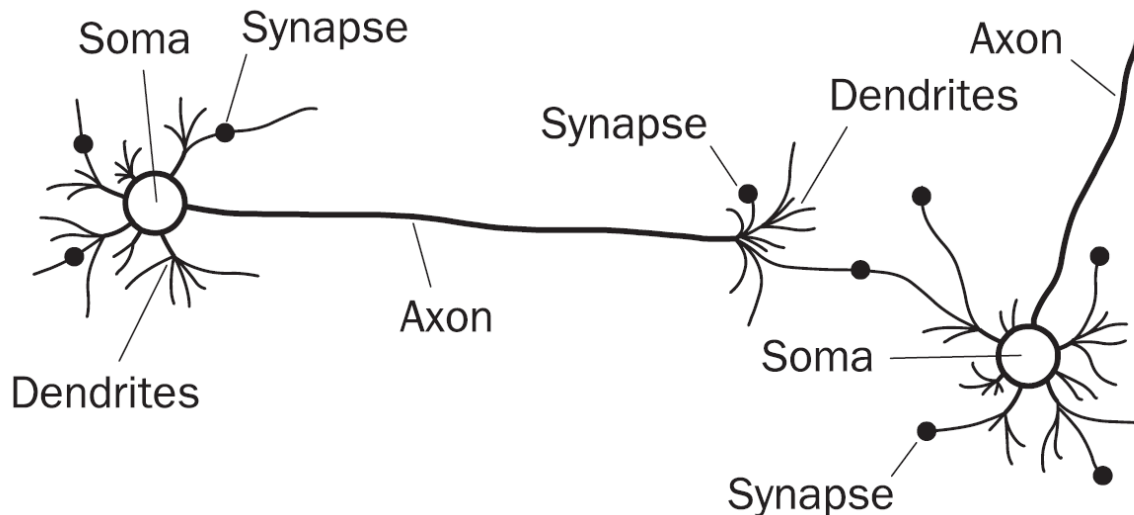


Figure II.4. Biological Neural Network [47]

II.5.2. Artificial Neural Network

Artificial neural networks, or ANNs, are computational models that draw inspiration from the complex web of neurons in a mammal's nervous system. These networks are composed of layers of interconnected neurons that communicate with each other to perform computations.

The inception of ANNs can be traced back to the 1950s with the introduction of the perceptron model, followed by the development of the back-propagation algorithm in the late 1960s. Some scholars suggest that the origins of these techniques might be even older.

The study of neural networks was a major focus of research until the 1980s. However, the dawn of the 21st century saw a resurgence in interest, spurred by the advent of rapid learning algorithms, the availability of Graphics Processing Units (GPUs) for computation, and the abundance of data. This renewed interest paved the way for the evolution of Deep Learning, characterized by networks with over 200 layers.

Chapter II: State of The Art - Machine Learning & Deep Learning

The structure of deep learning networks mirrors the layered organization of the human visual system. The visual cortex, V1, is responsible for basic image properties, with billions of connections. Subsequent visual areas like V2, V3, V4, and others process more complex concepts like shapes and faces. Deep learning draws inspiration from this hierarchical visual system organization [48]. Figure II.5 displays a schematic drawing of an Artificial Neural Network.

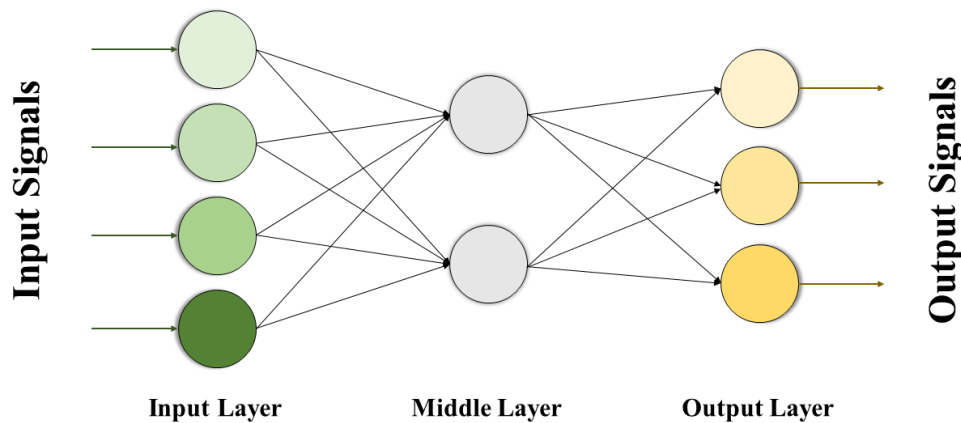


Figure II.5. Artificial Neural Network [47]

II.5.2.1. Multi-Layer Perceptron

A multilayer perceptron is among the methods in machine learning used to solve various problems. It has a reasonably simple and understandable structure. MLP must have at least three layers, the input data, weights, and biases, and an activation function. The neurons of one layer transmit the output to the next layer's neurons with the help of the adaptive weight coefficients. After each neuron's output is multiplied by the weight coefficient, a non-linear activation function is applied to get its final value, often as a sigmoid or hyperbolic tangent. Weights are then re-evaluated using an error function at the end of the training process, and the weights are multiplied by the learning rate and the error in each epoch. The process is repeated until the calculated weights are the last, the number specified before training [49].

II.6. Deep Learning

Deep learning, being a subset within the realm of machine learning and artificial intelligence, leverages deep, multilayered artificial neural networks. Numerous advancements in artificial intelligence can be attributed to this field. The efficacy of deep learning is most pronounced when tackling intricate issues, although it can also handle simpler classification tasks [50]. In specific domains, deep learning has empowered machines to rival or sometimes exceed human capabilities [45].

II.6.1. Convolutional Neural Networks (CNNs)

A convolutional neural network (CNN) is a specialized form of neural network designed for image analysis, with the capability to also process audio and text data. In contrast to fully connected (dense) layers, where each neuron connects with all neurons in the preceding layer, CNNs arrange neurons in three dimensions: height, width, and depth. Through the utilization of convolutional layers, CNNs can detect local patterns like textures and edges, resulting in reduced parameter size and mitigated risks of overfitting compared to dense layers [45].

CNNs demonstrate notable efficacy in image classification due to their spatial consciousness, mirroring the hierarchical organization of the visual cortex in the human brain. This enables them to interpret data ranging from individual pixels to intricate characteristics such as objects and facial features. These networks have quickly become a disruptive technology, setting new performance benchmarks in various domains beyond image processing [48].

II.6.1.1. Convolutional Neural Networks architecture

The architecture of the Convolutional Neural Network consists of multiple layers responsible for extracting and acquiring significant features from the input data, thereby facilitating classification through the efficient representation of the input data. Figure II.6 illustrates a depiction of a Convolutional Neural Network Architecture.

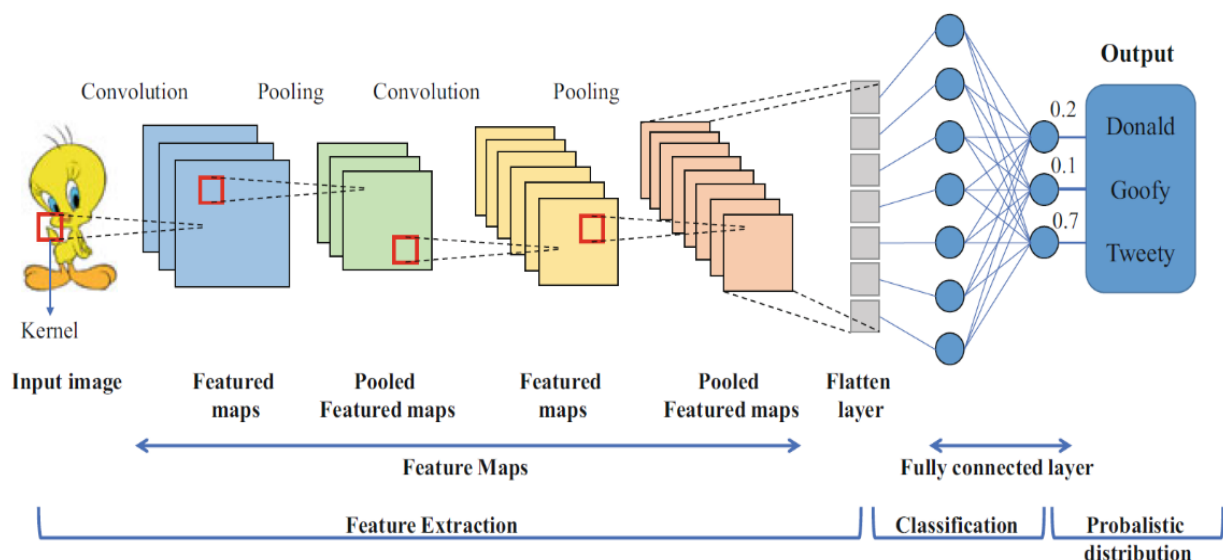


Figure II.6. Convolutional Neural Networks architecture [44]

Chapter II: State of The Art - Machine Learning & Deep Learning

- **Convolution layer**

The fundamental components of CNNs are the convolutional layers, which are responsible for applying learnable filters (referred to as kernels) to the input data through convolution operations. These filters are designed to identify particular features or structures within the input, such as edges, textures, or shapes. Through the utilization of multiple filters, the network is capable of acquiring hierarchical representations that exhibit greater levels of complexity, the convolution operation is illustrated in Figure II.7 [44].

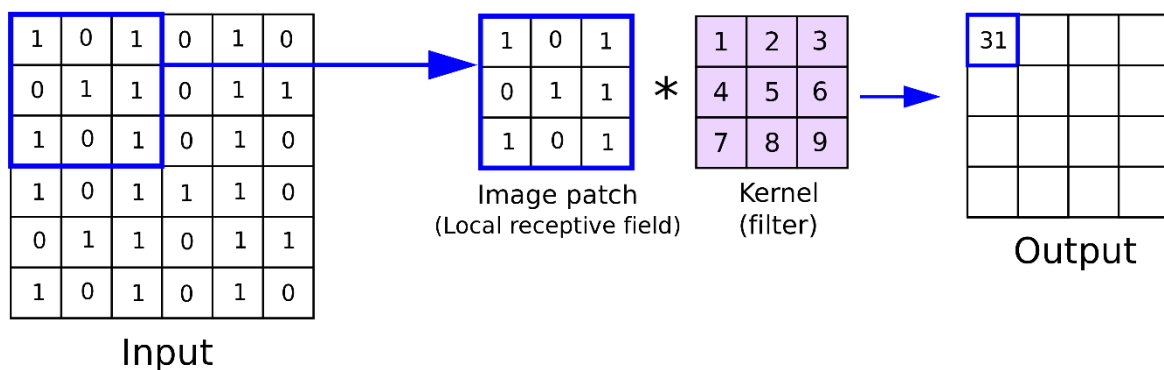


Figure II.7. Convolution operation [51]

- **Pooling layer**

The pooling layer plays a crucial role in subsampling the feature maps generated by convolutional operations. Its main goal is to reduce the size of the feature maps while retaining the significant features at each pooling stage. The pooling operation involves assigning stride and kernel sizes, and there are various types of pooling methods available, including max, min, and GAP pooling. While Figure II.8 demonstrates these three techniques, other methods such as tree pooling, gated pooling, and average pooling can be used.

However, it is worth noting that the pooling layer may sometimes decrease the overall performance of the CNN model. This is because it focuses solely on identifying specific features' accurate locations and may overlook other essential information [52].

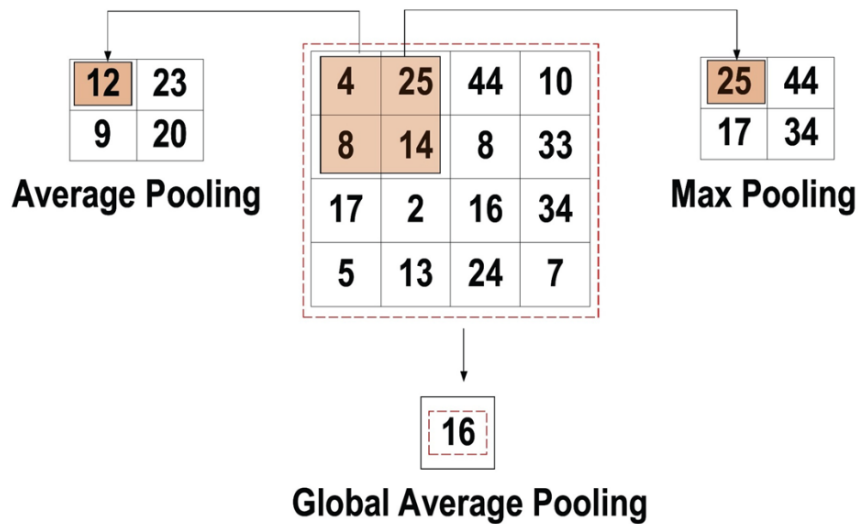


Figure II.8. Three types of pooling operations [52]

- **Flatten layer**

The flattening layer transforms a two-dimensional matrix into a one-dimensional vector before the input of the data into the fully connected dense layer [45].

- **Fully connected layer**

Adding a Fully Connected layer (also known as a Dense layer) allows learning non-linear combinations of high-level features from the convolutional layer. The fully connected layer learns non-linear functions in that space. Flattening an image into a column vector prepares it for a Multi-Level Perceptron. The flattened output is used in a feed-forward neural network with backpropagation during training. Through epochs, the model can distinguish between different features in images and classify them using Softmax Classification, which assigns probabilities to each class and selects the most likely class for a given input [53].

- **Dropout layer**

Dropout, a common regularization method, is employed in neural networks to combat overfitting. Overfitting in CNNs often occurs due to a high number of trainable parameters. A pre-set dropout rate determines the likelihood of a neuron being excluded during training, addressing overfitting. This rate is referred to as the dropout rate [41].

II.6.1.2. Activation function

- **ReLU function**

ReLU (Rectified Linear Unit), setting negative values to zero. ReLU helps the output of a convolutional layer contain only positive values, aiding learning and reducing vanishing gradients (Figure II.9). ReLU is popular in CNNs for its simplicity, efficiency, and accuracy improvement on various tasks.

The function is defined by:

$$f(x) = \max(0, x) \tag{II.1}$$

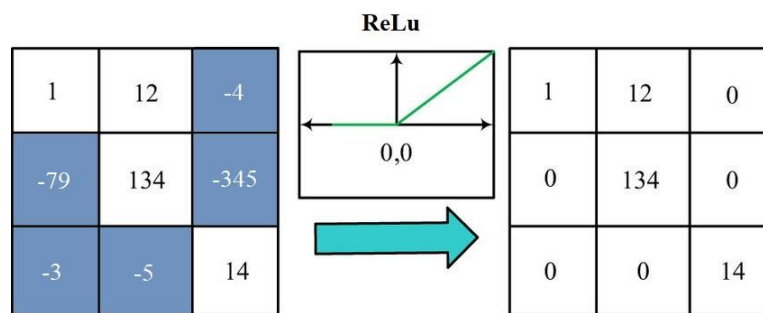


Figure II.9. ReLU activation function operation process [54]

- **Softmax Function**

In the field of machine learning, the Softmax function is a commonly employed mathematical function that converts numerical values into a probability distribution. The function generates a vector that signifies the probability distributions of various potential outcomes. This method is utilized to standardize the network's output into a probability distribution among predicted output classes [55], [56].

The function is defined by:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}} \tag{II.2}$$

Categorical Cross-Entropy Loss Function

The categorical cross-entropy loss function, also known as Softmax loss, is commonly used for multi-class classification problems. It measures the performance of a classification model whose output is a probability value between 0 and 1.

The function is defined by:

$$\text{Loss} = - \sum_{i=1}^n \sum_{c=1}^C y_{i,c} \log(\hat{y}_{i,c}) \tag{II.3}$$

II.6.1.3. CNN Models

Numerous CNN models have been developed by scholars in previous studies. These models exhibit varied architectures characterized by differences in layer dimensions, the overall quantity of layers, types of layers, and interconnections among layers [57]. Some of the most popular models include AlexNet, VGGNet, GoogleNet, ResNet, and EfficientNet (Figure II.10).

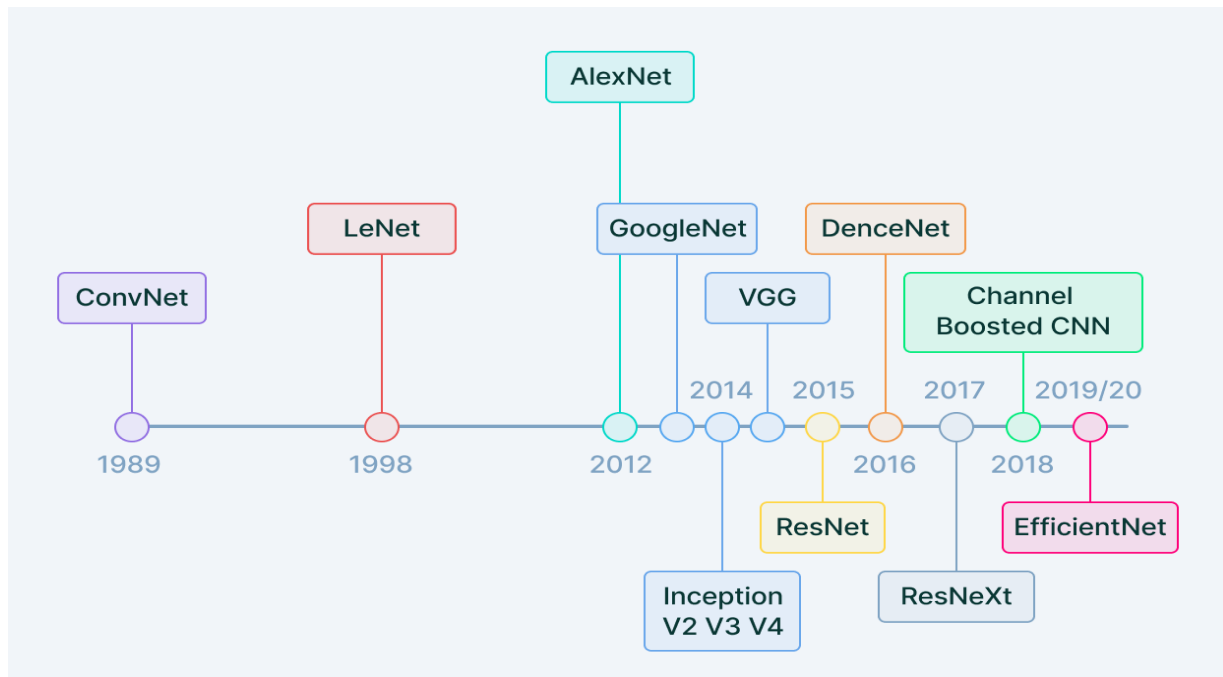


Figure II.10. Timeline of CNN models [58]

Lightweight and performance-efficient inference models have also been deployed applications in resource-constrained devices through the utilization of software libraries and Application Programming Interfaces (APIs). These libraries/APIs incorporate a variety of kernels and quantization techniques aimed at minimizing the memory and computational demands of conventional machine learning (ML) models [59].

▪ MobileNetV2

MobileNetV2 is a convolutional neural architecture design renowned for its efficient performance on mobile devices. It is based on a suboptimal residual architecture, where the primary connections exist between the bottleneck layers. The central expansion layer channels serve as a form of non-linearity through the utilization of lightweight depth-wise convolutions. In essence, the structure of MobileNetV2 consists of an initial 32-channel fully convolutional layer, succeeded by 19 residual bottleneck layers [60] (Figure II.11). MobileNet is a favorable

Chapter II: State of The Art - Machine Learning & Deep Learning

option when seeking an equilibrium between a sophisticated, high-performing deep neural network and efficiency [61].

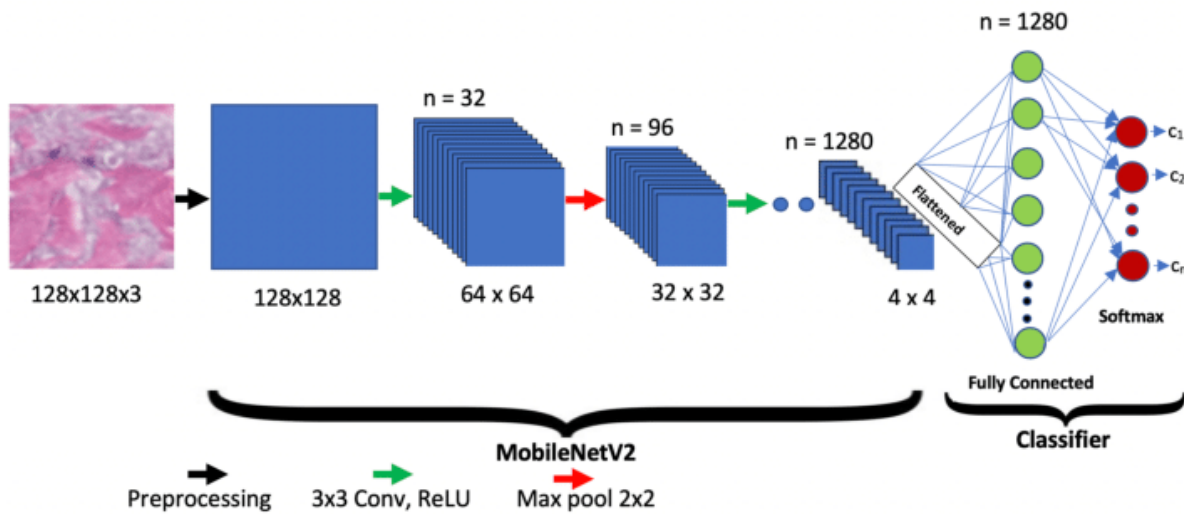


Figure II.11. The MobileNetV2 network architecture [62]

II.6.1.4. Advantages and Disadvantages of Convolutional Neural Networks

- **Advantages**

- A notable advantage of this approach is its high level of precision.
- This method is specifically crafted to handle visual data.
- It can hierarchically capture spatial characteristics [63].

- **Disadvantages**

- Requires large amounts of labeled data.
- High computational costs.
- Optimizing this model can be challenging due to the extensive parameter scale [63].

II.6.2. Transfer Learning

Based on our collective expertise, individuals possess the capacity to acquire novel expertise with ease. Our aptitude for learning is heightened, particularly when the current task aligns with our prior experiences. To illustrate, mastering a new programming language for a computer expert or operating a different kind of vehicle for an experienced driver is notably uncomplicated, drawing from our established knowledge.

Chapter II: State of The Art - Machine Learning & Deep Learning

Transfer learning is a domain within Machine Learning that strives to leverage the acquired insights from addressing a particular issue to tackle a distinct yet interconnected problem [64].

Transfer learning occurs when knowledge or skills obtained in one specific area, referred to as the source, are utilized in another region, known as the target. An illustration of this concept is the application of a CNN which has been trained on a substantially labeled dataset, to a different dataset that connects with the original training data. This practice holds particular significance in image analysis, especially when CNNs trained on the extensive ImageNet database are directly employed in other image analysis tasks within a specific domain, or are retrained using relatively small datasets with minimal adjustments from this new domain [60]. Figure II.12 illustrates the difference between Traditional machine learning and Transfer learning.

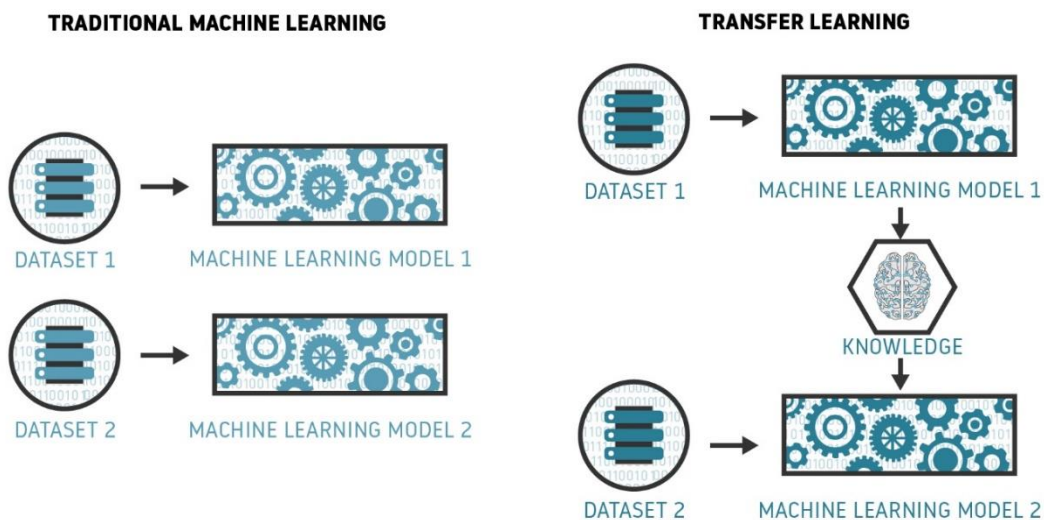


Figure II.12. Traditional machine learning vs Transfer learning [65]

II.6.2.1. How Transfer Learning Works

- **Pre-trained Model:** Start by utilizing a model that has undergone prior training for a specific task with a vast amount of data. Often trained on substantial datasets, this model has recognized overarching characteristics and trends applicable to various associated tasks.
- **Transfer Layers:** Refer to a group of layers identified in the pre-existing model, which are responsible for capturing fundamental information that applies to both the new task at hand and the original task. These particular layers are inclined towards acquiring basic details and are commonly situated close to the uppermost part of the network.

Chapter II: State of The Art - Machine Learning & Deep Learning

- **Fine-tuning:** Involves the utilization of the dataset obtained from the recent challenge to retrain the selected layers, a process that is explicitly labeled as fine-tuning. The primary objective of this approach is to maintain the existing knowledge acquired during pre-training, thereby allowing the model to adjust its parameters to align more effectively with the requirements of the present task [66].

II.6.3. Transformers

Transformers represent a category of Deep Learning models that brought about a paradigm shift in the field of natural language processing (NLP) by supplanting traditional recurrent neural network designs such as LSTMs. In contrast to LSTMs, which sequentially process elements within a sequence, transformers leverage an attention mechanism to enhance the handling of sequential data, leading to heightened efficiency and remarkable achievements in NLP assignments.

The proliferation of transformer models has exerted a profound influence on machine translation (MT), ushering in a new era of Neural Machine Translation (NMT) driven by the capabilities of transformers. Noteworthy are the multilingual transformer models that have surfaced to tackle the complexities inherent in language processing tasks, thereby influencing the evolution of expansive language models (LLMs) and attention mechanisms across a spectrum of applications [67].

Expanding beyond the realm of NLP, transformers have made inroads into the domain of image processing through innovative architectures like the Vision Transformer (ViT). The ViT framework dissects images into smaller patches, incorporates positional embeddings, and applies a conventional transformer encoder to process the information. While originally conceived for NLP purposes, transformers have demonstrated significant performance gains over conventional convolutional neural networks (CNNs) particularly in scenarios involving vast datasets, underscoring their adaptability across diverse domains beyond conventional language-centric activities [68].

The foundation of the transformer architecture resides in the concept of self-attention, which serves as the fundamental building block of transformer models. Self-attention enables the model to evaluate the importance of various elements in the input sequence about each other, leading to the creation of contextualized representations that encompass dependencies throughout the entire sequence. Through this mechanism, transformers are capable of concurrently processing input data, thus enhancing their scalability and efficiency.

Chapter II: State of The Art - Machine Learning & Deep Learning

The architecture of the transformer comprises a framework with an encoder-decoder structure, where each element consists of numerous layers of self-attention and feed-forward neural networks (Figure II.13).

During the processing of the input sequence, the encoder generates intricate contextual embeddings, while the decoder utilizes these embeddings to produce the output sequence incrementally.

Vision Transformers belong to a category of models that adapt transformer principles, initially designed for NLP tasks, to the realm of computer vision. The key components include:

- **Patch embedding:** Vision Transformers initiate by partitioning an input image into patches of fixed sizes, which are subsequently flattened and linearly transformed into embeddings, similar to tokens in NLP. This procedure converts the 2D spatial data of the image into a 1D token sequence suitable for transformer processing.
- **Positional encoding:** To preserve the positional details lost during patch embedding, Vision Transformers integrate positional encodings into the patch embeddings, ensuring the model can consider the position of each patch within the image.
- **Transformer encoder:** Comprising alternating layers of multi-head self-attention and feed-forward neural networks, the transformer encoder enables the model to evaluate the significance of different patches about each other, capturing both local and global connections.
- **Multi-head self-attention:** Through the utilization of multi-head self-attention, the transformer encoder enables the model to focus on diverse areas of the image concurrently, facilitating the capture of a wide range of features.
- **Feed-forward neural networks:** Positioned between self-attention layers, these networks apply additional transformations to the sequence of patch embeddings.
- **Layer normalization:** Within the encoder, normalization techniques are implemented to enhance learning stability and convergence.
- **Classification head:** Positioned at the apex of the transformer, a classification head (often a basic linear layer) is utilized to make predictions based on the encoded image representations. ViTs use components like self-attention to process images similar to how transformers handle sequential data. The scaled dot-product attention is crucial in the multi-head self-attention of transformers and ViTs.

Here is how it works:

Chapter II: State of The Art - Machine Learning & Deep Learning

- It compares each query with keys using dot product to determine the focus on values.
- Scores are scaled down to prevent small gradients in Softmax due to large key dimensions.
- Softmax function converts scaled scores into a probability distribution, reflecting the importance of each value based on query-key similarity.

The weighted sum of values using Softmax weights produces the output sequence with attention applied, incorporating information from other parts of the sequence [69].

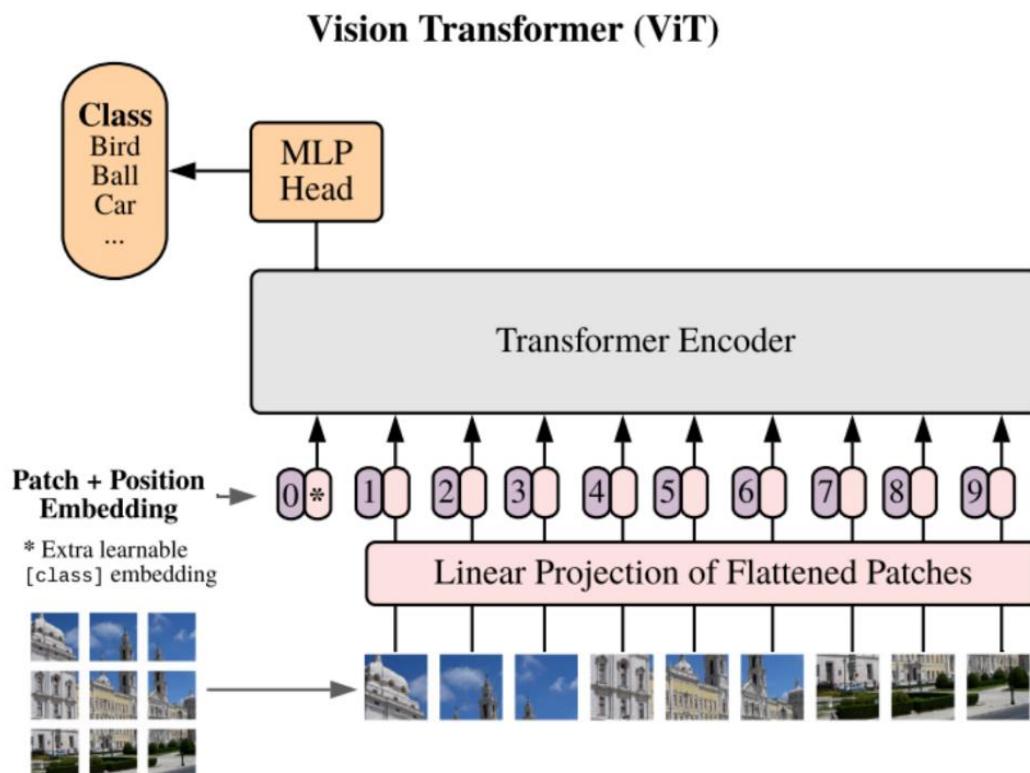


Figure II.13. Vision Transformer architecture [68]

II.7. Lightweight Models in Plant Disease Detection

This work focuses on lightweight models due to their significant advantages, including reduced computational requirements and faster processing times. These features make them particularly suitable for deployment on resource-constrained devices such as mobile phones and embedded systems. Table II.1 summarizes several lightweight models used in plant disease detection, highlighting their architectures, datasets, and achieved accuracies.

Chapter II: State of The Art - Machine Learning & Deep Learning

Author	Year	Algorithms	Dataset	Main findings	Memory Requirement
Anita Shrotriya, et al [70]	2023	lightweight Deep Convolutional Neural network model	Plant Village	97.73%	/
Guan, Fu et al [71]	2023	Dise-Efficient model	Plant Village	99.80%	/
Feng, Song et al [72]	2023	isotropic CNN model, FoldNet	Plant Village	99.84%	/
Wang, Zhang et al [73]	2023	Ultra-lightweight efficient network, ULEN	Plantvillage, Cassava,	98.13%, 54.97%	/
Sanida, Sanida et al [74]	2023	Lightweight CNN model	PlantVillage (Tomato)	99.04%	/
Fang, Zhen et al [75]	2023	Lightweight Multiscale CNN Model	LWDCD2020 (wheat)	98.7%	/
Verma, Kumar et al [76]	2023	Unified lightweight CNN model	Corn, Rice, and Wheat	The model achieved 99.74%, 82.67%, and 97.5% accuracy for Corn, Rice, and Wheat, respectively.	/
Huang, Wu et al [77]	2023	YOLOR-Light-v1, YOLOR-Light-v2, Mobile-YOLOR-v1, and Mobile-YOLOR-v2 models	PlantDoc	Achieved 60.4% mAP@ .5 in the PlantDoc	/
Liu, Song et al [78]	2023	NanoSegmenter model based on the Transformer structure	Collected dataset	98%	/
Thakur, Khanna et al [79]	2022	lightweight PlantXViT model combines CNN	Five datasets	Average accuracy of 93.55%, 92.59%, and 98.33% on Apple, Maize, and Rice	/

Chapter II: State of The Art - Machine Learning & Deep Learning

		with Vision Transformers		datasets, respectively.	
S. Wagle, et al. [80]	2021	compact convolutional neural networks (N1, N2, N3)	Plant Village	N1: 99.45% N2: 99.65% N3: 99.55%	N1, N3: 14.8 MB N2: 29.7 MB
Yang Liu, et al. [81].	2021	CNN based on SqueezeNet	Plant Village	98.46%	0.62 MB
Muhammad Hammad Saleem, et al. [82]	2020	SSD, RCNN, and RFCN	Plant Village	73.07%	/

Table II.1. Lightweight models in plant disease detection

II.8. Conclusion

This chapter has explored some of the most cutting-edge and dependable methodologies in machine learning (ML) and deep learning (DL). Specifically, we focused on the most commonly used lightweight models for plant disease detection within these fields.

The subsequent chapter will discuss the hyperparameters and the evaluation metrics.

Chapter III

Fine-tuning and Evaluation

Metrics

Chapter III: Fine-tuning and Evaluation Metrics

III.1. Introduction

In the previous chapters, we reviewed the state-of-the-art advancements in plant disease detection using deep learning.

This chapter delves into fine-tuning deep learning models for detecting plant diseases, emphasizing key hyperparameters for CNNs, Transformers, and LightGBM. It also discusses various evaluation metrics like confusion matrices, accuracy, precision, recall, and F1 score to gauge the performance of classification models with imbalanced data.

III.2. Hyperparameters

Hyperparameters are parameters controlling model structure and learning process. They are top-level parameters. Examples include train-test split ratio, learning rate, activation function choice, cost function, number of layers and units, dropout rate, iterations, kernel sizes, pooling size, and batch size. Hyperparameters are classified into different categories.

III.2.1. CNN hyperparameters

a) Hyperparameters related to network structure

1. Number of hidden layers

Determines the profundity of the neural network. Increased layers can capture more intricate patterns but also elevate the likelihood of overfitting and computational cost.

2. Number of activation units (Neurons) in each layer

Specifies the width of the network. Augmented neurons can enhance the network's capability to glean insights from data, yet they also contribute to the computational load.

3. Choice of activation function

Typical options comprise Sigmoid and ReLU. The choice of activation function impacts the network's learning capability and its problem-solving aptitude. For instance, ReLU is extensively utilized due to its effectiveness in mitigating the vanishing gradient problem.

4. Kernel or filter sizes in convolutional layers

Chapter III: Fine-tuning and Evaluation Metrics

The size of the filters (e.g., 3x3, 5x5) employed in the convolutional layers. This factor influences the receptive field of neurons and the specificity of feature detection.

5. Pooling size

The size of the pooling window (e.g., 2x2) utilized in pooling layers. Pooling assists in reducing the spatial dimensions of the input, thereby decreasing the computational burden and helping to improve the translation invariance of the representation.

6. Dropout rate (Dropout probability)

A regularization method to forestall overfitting by randomly setting a portion of input units to 0 during each update in the training phase.

b) Hyperparameters related to the training process

1. Train-Test split ratio

Determines the ratio of the dataset allocated for training versus testing. A prevalent split is 80% for training and 20% for testing, although this distribution can vary based on the dataset and the task. In our case, we use 80% for training, 10% for testing, and 10% for validation.

2. Learning rate

Regulates the step size in the gradient descent optimization process. A higher learning rate can expedite training but may lead to suboptimal convergence or divergence. Conversely, a lower learning rate ensures more consistent convergence but can make the training process slow.

3. Choice of optimization algorithm

Diverse techniques such as Stochastic Gradient Descent (SGD), Adam, and RMSprop can be employed. Each technique adjusts the learning rate and gradient handling uniquely, impacting the speed and quality of convergence.

4. Choice of cost or loss function

Determines how the performance of the network is measured. Common loss functions include Mean Squared Error (MSE) for regression tasks and Cross-Entropy Loss for classification tasks.

5. Number of iterations (Epochs)

The number of times the entire dataset is processed by the network during training. More epochs can enhance learning but also heighten the risk of overfitting.

6. Batch size

The number of training instances utilized in a single iteration. A larger batch size can offer more consistent gradient approximations but necessitates more memory, whereas a smaller batch size can introduce more noise in training while demanding less memory and often yield better generalization.

Batch Size is among the important hyperparameters in Machine Learning. It specifies the samples processed before updating model parameters. It is essential for maximizing model performance [83], [84].

III.2.2. Transformers hyperparameters

a) Hyperparameters related to network structure

- 1. Number of encoder/decoder layers:** Defines the depth of the transformer network. More layers capture complex patterns but increase computational cost and overfitting risk.
- 2. Hidden size (`d_model`):** Specifies the dimensionality of input and output vectors. Larger sizes improve capacity but add computational load.
- 3. Number of attention heads:** Number of parallel attention mechanisms per layer. More heads improve the model's ability to focus on different input parts.
- 4. Feed-forward network size:** The number of units in the inner feed-forward network is typically larger than the hidden size to ensure learning capacity.
- 5. Activation function:** Common choices include ReLU and GELU, affecting learning ability and performance.
- 6. Dropout rate:** Probability of dropping units during training to prevent overfitting, applied to attention weights and feed-forward outputs.

b) Hyperparameters related to the training process

- 1. Learning rate:** Controls the optimization step size. Higher rates speed up training but risk overshooting; lower rates stabilize but slow down training.
- 2. Batch size:** Number of samples per iteration. Larger sizes stabilize gradients but need more memory; smaller sizes introduce noise that aids generalization.

Chapter III: Fine-tuning and Evaluation Metrics

- 3. Warm-up steps:** Initial steps with linearly increasing learning rate to stabilize early training.
- 4. Optimizer choice:** Algorithms like Adam, AdamW, or RMSprop, each impacting training speed and convergence quality differently.
- 5. Weight decay:** The regularization technique penalizes large weights to prevent overfitting.
- 6. Gradient clipping:** Limits gradient magnitude to prevent exploding gradients.
- 7. Label smoothing:** Prevents overconfidence by slightly altering target labels, and improving generalization.
- 8. Sequence length (Max position embeddings):** Maximum input sequence length the model can handle, balancing context handling and computational complexity.
- 9. Number of epochs:** Total passes through the training dataset. More epochs improve learning but increase overfitting risk.
- 10. Cost or loss function:** Measures performance and guides optimization. Choices include Cross-Entropy Loss for classification and Mean Squared Error (MSE) for regression.

III.2.3. LightGBM hyperparameters

a) Core parameters

- 1. Objective:** Defines the type of learning task, in our case, multiclass classification.
- 2. Boosting type:** The boosting algorithm to use is the Gradient Boosting Decision Tree (gbdt).
- 3. Num class:** Specifies the number of classes in the multiclass classification problem.
- 4. Metric:** The metric used to evaluate the model's performance, set to multi_logloss.

b) Learning control parameters

- 1. Learning rate:** The learning rate controls the step size during gradient descent.
- 2. Num leaves:** The maximum number of leaves in one tree, which controls the complexity of the model.
- 3. Verbose:** Controls the verbosity of the training process. A value of -1 suppresses most of the output [85].

III.3. Evaluation metrics

Assessment measures are employed to evaluate the efficacy of the model. An essential aspect within the realm of deep learning pertains to the methodology utilized for model

Chapter III: Fine-tuning and Evaluation Metrics

evaluation. The assessment of the model's predictive accuracy stands out as a fundamental factor to consider in the model's developmental stage. Inadequate model assessment and improper employment of assessment measures can lead to erroneous predictions, particularly in scenarios involving imbalanced datasets. Therefore, it is advisable to utilize a diverse range of evaluation metrics, as outlined below [48].

III.3.1. Confusion matrix

In classification, data points are labeled and compared to predicted classes. Results are categorized into true positive, true negative, false positive, and false negative. These values are used for evaluation metrics in classification tasks and are usually shown in a confusion matrix table [48] (Table III.1).

		True Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

Table III.1. Confusion matrix.

III.3.2. Accuracy

Accuracy is an evaluation metric for classification models. It is defined as the number of correct predictions divided by the total number of predictions. Accuracy is not the ideal metric in situations where you have imbalanced datasets.

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Positive} \quad (III.1)$$

III.3.3. Precision

Precision is characterized as the quotient of true positives to the sum of true positives and false positives. It indicates the frequency with which the model makes accurate

Chapter III: Fine-tuning and Evaluation Metrics

predictions when, they are positive. Precision is a valuable metric in situations with high repercussions of false positives.

$$P = \frac{TP}{TP+FP} \quad (\text{III.2})$$

III.3.4. Recall

Recall is a valuable metric to employ when the repercussions of false negatives are significant. It is characterized as the quotient of true positives divided by the sum of true positives and false negatives.

$$R = \frac{TP}{TP+FN} \quad (\text{III.3})$$

III.3.5. F1 score

The F1 measure or F1 score combines precision and recall. It is used to evaluate multiclass classifiers, especially with uneven class distribution. F1 score ranges from 0 to 1. A good F1 measure indicates low false negatives and false positives. The formula for F1 measure is:

$$F1_Measure = 2 * (precision * recall) / (precision + recall) [48] \quad (\text{III.4})$$

III.4. Conclusion

This chapter explained the optimization of deep learning models for plant disease detection. It emphasizes critical hyperparameters tailored for CNNs, Transformers, and LightGBM, alongside key evaluation metrics like confusion matrices, accuracy, precision, recall, and F1 score. These metrics are pivotal for evaluating classification model performance, especially in scenarios involving imbalanced data.

The following chapter will delve into a case study, presenting our proposed solution and outlining the design of our application.



Chapter IV

Design, Implementation
& Results

Chapter IV: Design, Implementation & Results

IV.1. Introduction

In the previous chapters, we explained the different hyperparameters of CNNs, Transformers, and LightGBM, as well as key evaluation metrics for said architectures.

This chapter focuses on the practical implementation of a plant disease detection system, utilizing three deep learning models: MobileNetV2, Vision Transformer (ViT), and AgriNetBoost. We detail the integration of these models into a web-based application and present the results through validation examples. A comparative analysis is conducted to determine the most effective model for plant disease classification.

IV.2. Working environment

The computational hardware employed while building the model comprises a personal computer with a Linux (Ubuntu) operating system, equipped with an i9 12900KF central processing unit, an Nvidia RTX 3090 graphics processing unit, and 64 Go of RAM DDR4. The primary programming language utilized is Python within the Anaconda environment, leveraging the Jupyter Notebook compiler and supplemented by using Google Colab for certain tasks. Moreover, for the development and validation stages of the application, a laptop computer running Windows 10 64-bit operating system, powered by an Intel (R) Core (TM) i5-9300H CPU at 2.40 GHz and 24GB of RAM, is utilized. Python is used as the programming language in conjunction with the Streamlit framework.

IV.3. Presentation of the languages used

The model under development pertains to a computer vision task (image classification). The primary tools utilized in the implementation process include Python as the main programming language and Streamlit for app deployment. This aligns with the prevalent practice of leveraging Python and key libraries such as TensorFlow, Keras, Scikit-learn, NumPy, PyTorch, LightGBM, Transformers, ONNX, and Streamlit in the development of deep learning models.

IV.3.1. Introducing the Python language

Python is an object-oriented, high-level programming language that provides developers with a versatile tool to efficiently and swiftly carry out tasks while promoting code readability and maintainability. It offers a wide range of libraries and frameworks that allow users to seamlessly integrate various systems, enhancing productivity, and facilitating the development of robust and sophisticated software solutions. (python website).



IV.3.2. Introducing the imported libraries

- **TensorFlow:** Developed by members of Google Brain's Machine Intelligence team, is a versatile open-source platform designed for machine learning and neural network research. Beyond its original purpose, TensorFlow has become widely adaptable, finding applications across various domains. Within its ecosystem, TensorFlow offers a plethora of tools, libraries, and community resources, plus stable Python and C++ APIs and backward-compatible APIs for multiple programming languages. These resources empower researchers to push the boundaries of machine learning and enable developers to easily build and deploy machine learning-driven applications that can seamlessly adapt to any computing environment [86].
- **Keras:** keras is a Simple, Flexible, Powerful multi-backend deep learning API, written in Python and capable of running on top of either JAX, TensorFlow, or PyTorch. The benefits include achieving optimal performance for models, maximizing ecosystem compatibility, and increasing the distribution of open-source models. Furthermore, Keras supports data pipelines from different sources, offering flexibility in training models [87].
- **LightGBM:** Is a framework for gradient boosting that employs algorithms based on trees. It has been engineered to exhibit qualities of being both distributed and efficient, boasting advantages such as accelerated training speed and enhanced efficiency, reduced memory utilization, improved accuracy, and the ability to support parallel, distributed, as well as GPU learning. Furthermore, it demonstrates capability in managing data of large scales [88].



Chapter IV: Design, Implementation & Results

- **Transformers:** the library transformers by Hugging Face and the community is used for cutting-edge Machine Learning in PyTorch, TensorFlow, and JAX. It offers numerous pre-trained models for tasks in text, vision, and audio across various modalities [89].



- **Pytorch:** Is a deep learning framework supported by the PyTorch Foundation, operating under The Linux Foundation, which fosters collaboration within the deep learning community, it engages in various activities to improve user experience and promote AI and deep learning tools within an open-source ecosystem. The Foundation's mission is to make advanced tools accessible to everyone [90].



PyTorch provides high-level features such as tensor computation with GPU acceleration and neural networks with an autograd system (short for automatic differentiation), simplifying the development of deep learning models [91].

- **ONNX:** Is an open ecosystem for AI developers to select tools as projects progress, offering a format for AI models and a computation graph model. It includes built-in operators and data type definitions. ONNX is well-supported in various frameworks, tools, and hardware, promoting interoperability and accelerating AI innovation. The community is encouraged to contribute and advance ONNX [92].



ONNX Runtime efficiently infers models from different frameworks (PyTorch, Hugging Face, TensorFlow) on various software and hardware. It utilizes hardware accelerators, supports multiple language APIs, and works on different devices and servers [93].

- **Streamlit:** Being a freely available and collaborative open-source Python framework, it serves as a valuable tool tailored specifically for individuals within the data science and artificial intelligence/machine learning engineering domains, offering them the capacity to create and distribute interactive data-driven applications that are characterized by their dynamic nature, all achieved through the utilization of a minimal amount of programming code. This framework empowers users to swiftly construct and launch robust data applications within a remarkably short span of time, exemplifying efficiency and effectiveness in the realm of data visualization and manipulation [94].



IV.4. Methodology

IV.4.1. System architecture

The architecture of the plant disease detection system (Figure IV.1) comprises a number of interconnected steps. Collecting data is initiated by utilizing the “PlantVillage” dataset, followed by thorough data preprocessing. A trio of models were created: pre-trained CNN (MobileNetV2), a Vision Transformer model (Google ViT) that was transformed into ONNX, and a hybrid model AgriNetBoost (LightGBM + MobileNetV2), founded on decision trees. The evaluation of the models was conducted based on specified performance metrics. The system has been incorporated with a Streamlit web application, facilitating instantaneous disease prognosis from downloaded images.

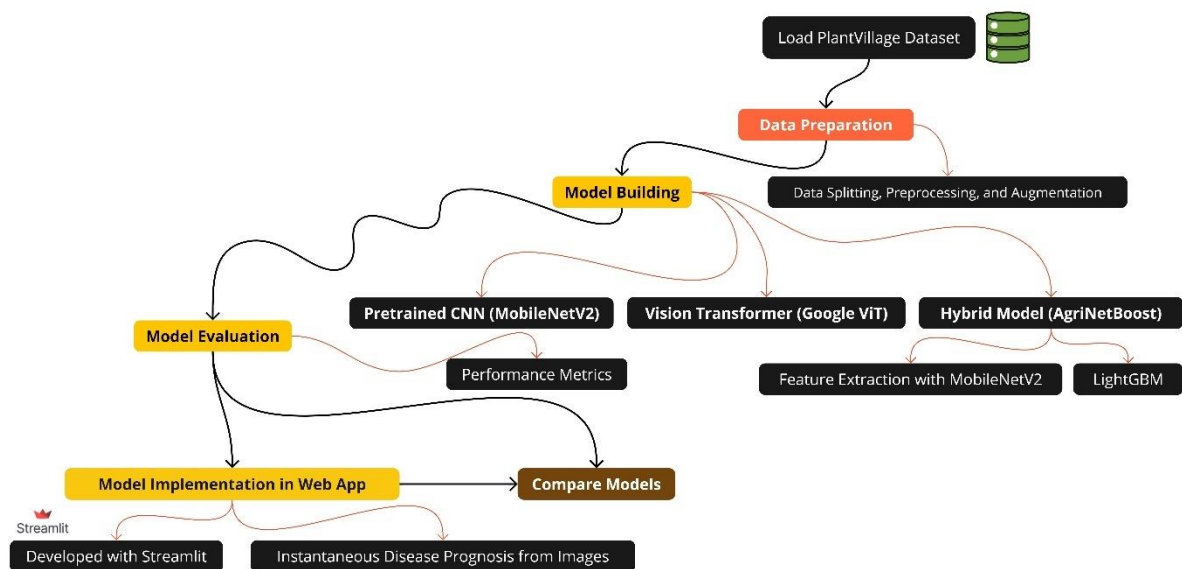


Figure IV.1. Detailed Workflow Diagram

IV.4.2. Used dataset

The dataset encompasses more than 50,000 photographs depicting both healthy and infected leaves across 14 different crop varieties, which include but are not limited to Apple, Blueberry, Cherry, Corn, Grape, Orange, Peach, Bell Pepper, Potato, Raspberry, Soybean, Squash, Strawberry, and Tomato. These images were procured at various research facilities tied to Land Grant Universities in the United States, such as Penn State, Florida State, and Cornell. Leaf samples were gathered by technicians from agricultural trials showcasing crops afflicted with specific ailments, utilizing a conventional digital camera (Sony DSC-Rx100/13 20.2

Chapter IV: Design, Implementation & Results

megapixels) to capture multiple images under diverse lighting scenarios. In instances of larger leaves, multiple segments were photographed. The dataset encompasses visuals of 17 fungal infections, 4 bacterial infections, 2 mold (oomycete) infections, 2 viral infections, and 1 infection induced by a mite, in addition to illustrations of healthy leaves pertaining to 12 crop varieties. The images underwent editing procedures involving background cropping and leaf orientation adjustments to ensure the apex pointed upwards. This dataset facilitates the advancement of machine learning algorithms for disease identification, with the objective of harnessing computer vision technology to combat crop yield diminishment resulting from plants diseases [95]. Figure IV.2 depicts a collection of leaf samples from various plants, each labeled with the respective disease or condition affecting them.

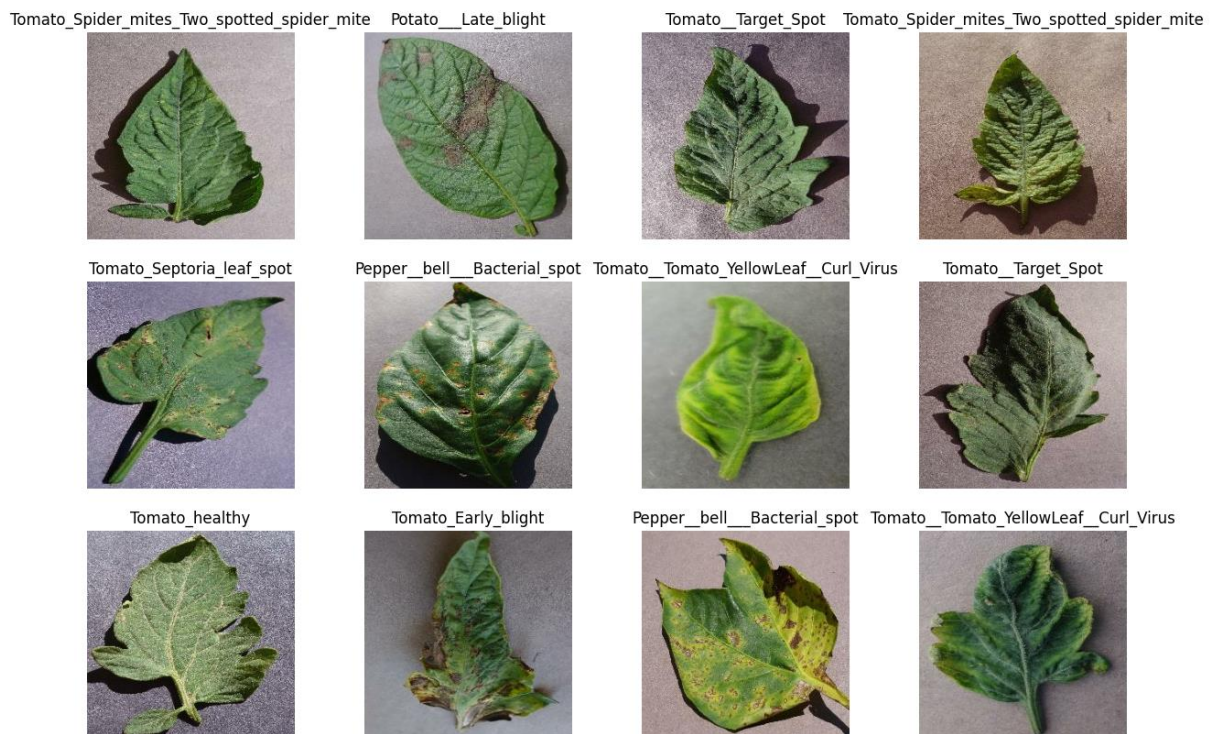


Figure IV.2. Visual dataset for identification and classification of plant diseases

IV.4.2.1. Dataset sample distribution

In this study, a subset dataset comprising 21,458 images representing three distinct plant species, namely bell pepper, potato, and tomato, was utilized. The analysis conducted involved a multi-class pathology examination of the infections affecting these three plant varieties. Specifically, pepper exhibited two distinct classes, potato demonstrated three classes, and tomato showcased ten classes. Details regarding the distribution of samples per class are provided in the table below.

Chapter IV: Design, Implementation & Results

Crops	Disease Type	Number of Images
Pepper	Bacterial spot	997
	Healthy	1478
Potato	Early blight	1000
	Late blight	1000
	Healthy	152
Tomato	Bacterial spot	2127
	Early blight	1000
	Late blight	1909
	Leaf Mold	952
	Septoria leaf spot	1771
	Spider mites Two-spotted spider mite	1676
	Target Spot	1404
	Yellow Leaf Curl Virus	3208
	Mosaic virus	373
	Healthy	1591
Total	15	21458

Table IV.1. Distribution of PlantVillage dataset samples

IV.4.2.2. Dataset advantages

- Diversity of Disease Representation: The dataset contains various plant diseases on tomato and pepper leaves. It facilitates thorough training of machine learning models to identify different types of diseases.
- Facilitation of Data Augmentation.
- Real-world Application.
- Educational Resource.
- Open source.

IV.4.2.3. Dataset disadvantages

- **Class Imbalance:** The dataset exhibits notable class imbalance due to certain diseases being underrepresented. Data augmentation can help address this issue but may not completely offset the lack of diverse, real-world examples for minority classes.
- **Potential for Overfitting:** Due to a scarcity of images, particularly for minority classes, there is a potential for overfitting, causing the model to focus on specific features of the training images rather than general patterns.
- **Scalability Issues:** In large-scale applications, expanding the dataset may be necessary. Acquiring and managing a bigger dataset can be resource and time consuming. This scalability problem can impede the dataset's widespread use in various regions and conditions.
- **Variability in Image Quality:** Variations in image quality, lighting conditions, and angles impact dataset consistency. Challenges in model training arise due to this variability, necessitating extra preprocessing for image normalization.

IV.4.3. Dataset preparation

IV.4.3.1. Dataset partitioning

A critical step in the process of data preparation involves partitioning the database into distinct sets designated for training, validation, and testing objectives. The distribution of data among these sets is enabled by the Scikit-learn package. In particular, 80% of the data is assigned to training, while 10% is allocated for validation and another 10% for testing, as illustrated in Table 05.

Chapter IV: Design, Implementation & Results

Crops	Disease Type	N°= of Images	Training	Validation	test
Pepper	Bacterial spot	997	997	797	100
	Healthy	1478	1478	1182	147
Potato	Early blight	1000	1000	800	100
	Late blight	1000	1000	800	100
	Healthy	152	152	121	15
Tomato	Bacterial spot	2127	2127	1701	212
	Early blight	1000	1000	800	100
	Late blight	1909	1909	1527	190
	Leaf Mold	952	952	761	95
	Septoria leaf spot	1771	1771	1416	177
	Spider mites Two-spotted spider mite	1676	1676	1341	167
	Target Spot	1404	1404	1123	140
	Yellow Leaf Curl Virus	3208	3208	2566	321
	Mosaic virus	373	373	298	37
	Healthy	1591	1591	1272	159
Total	15	21458	21458	16505	

Table IV.2. Distribution of PlantVillage dataset samples

IV.4.3.2. Data preprocessing

The preprocessing conducted in this research encompasses the actions of resizing and normalizing the data:

- Resizing was carried out to address the variability in image sizes within the dataset. The images were resized to a standardized dimension of 224x224 for MobileNetV2 and AgriNetBoost, and to 256x256, which is the original size of the dataset images.
- In terms of normalization, the pixel values underwent normalization by being divided by 255, which ensured that the pixel intensities fell within the range of 0 to 1.

These preprocessing procedures played a vital role in ensuring the compatibility of the images with the models utilized in the study.

IV.4.3.2. Data augmentation

Imbalance within the dataset is evident, characterized by an uneven representation of classes where some have more instances than others, it is essential to accurately identify minority classes. To ensure that all classes are taken into account equally, and not just the majority. This requires an increase in the volume of data for each class. A common method known as “data augmentation” is widely adopted to improve the training dataset by incorporating modified or synthetic data derived from the existing dataset. The following section develops this technique.

- **Random crop:** Throughout the training process, this particular process will engage in a random selection of a specific location for cropping images to a desired target size. Moreover, it is important to note that all images within a given batch will undergo cropping at an identical location [96].
- **Random flip:** This stratum will horizontally or vertically flip the images depending on the mode attribute. In the period of inference, the resulting output will remain indistinguishable from the initial input [96].
- **Random translation:** During the training process, this particular layer is responsible for introducing random translations to each image. It accomplishes this task by utilizing various factors such as height, width, fill mode, interpolation, fill value, and data format to fill any empty space within the image [96].

Chapter IV: Design, Implementation & Results

- **Random rotation:** will implement arbitrary rotations to individual images, thereby occupying vacant areas. By default, arbitrary rotations are exclusively employed throughout the training phase. During the inference period, it remains inactive [96].
- **Random zoom:** randomly zooming images during training, this component adjusts images by zooming in or out independently on each axis and filling any empty space according to the specified fill mode. Its behavior is controlled by several arguments height_factor, width_factor fill_mode etc [96].
- **Random contract:** will arbitrarily modify the contrast of an image or multiple images through a stochastic factor. Contrast is altered autonomously for every channel of each image while undergoing training. In the case of each channel, it calculates the average of the image pixels within the channel and subsequently modifies each element of every pixel.
- **Random brightness:** will arbitrarily augment/diminish the luminosity of the input RGB images. During the inference stage, the resulting outcome will mirror the initial input.

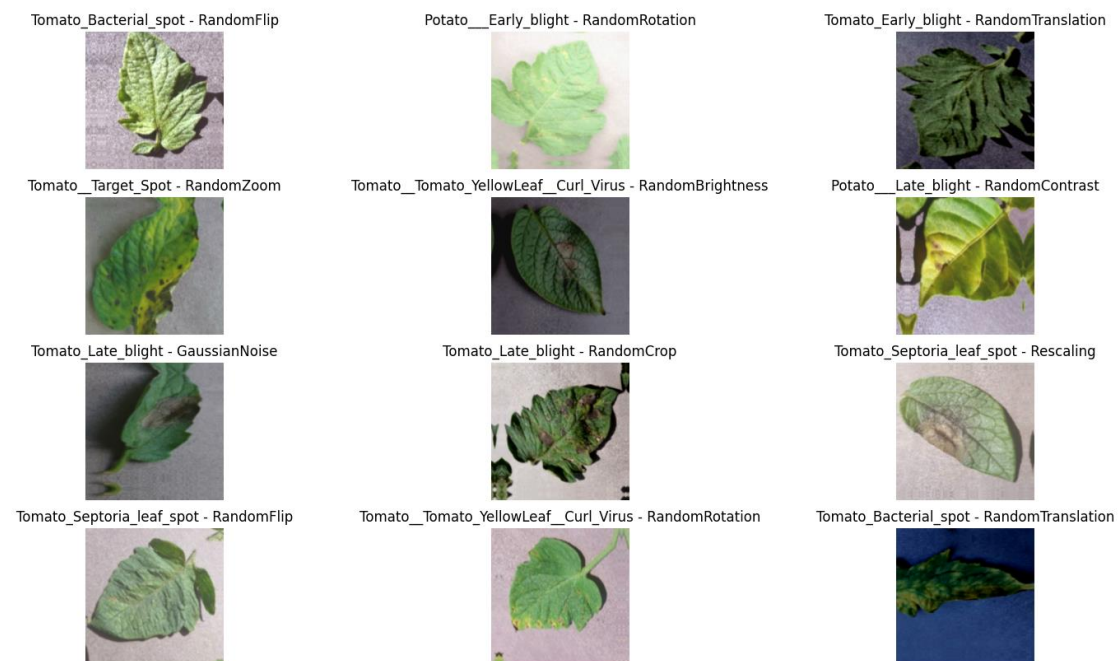


Figure IV.3. Visualization of data augmentation

IV.4.4. Used models

Our system for detecting plant diseases has been created through the design of multiple pre-trained models and a hybrid model, which incorporates a degree of fine-tuning tailored to the specific study framework.

1) MobileNetV2

MobileNetV2 is founded on the principles of depth-wise separable convolutions, which were first introduced in its predecessor MobileNetV1, accompanied by additional enhancements. The fundamental elements of MobileNetV2 comprise:

- **Depthwise Separable Convolutions:** This procedure segregates the spatial and channel-wise convolutions, leading to a notable reduction in computational complexity and parameter count in contrast to standard convolutions.
- **Inverted Residuals with Linear Bottlenecks:** MobileNetV2 presents inverted residual blocks, featuring a slim intermediate layer (bottleneck layer) succeeded by an expansion layer. This configuration aids in upholding model efficiency and effectiveness.
- **ReLU6 Activations:** To avert the issue of vanishing gradients, MobileNetV2 adopts ReLU6 activations as opposed to conventional ReLU.

The model used in this study is a pre-trained MobileNetV2, fine-tuned using the PlantVillage dataset. We used MobileNetV2 as the base model with pre-trained weights from ImageNet, keeping its layers non-trainable. On top of this, we added a GlobalAveragePooling2D layer, followed by dropout layers, a dense layer with ReLU activation, another dropout layer to mitigate overfitting, and a final dense layer with Softmax activation to classify the plant disease categories. The model architecture is illustrated in Figure IV.4.

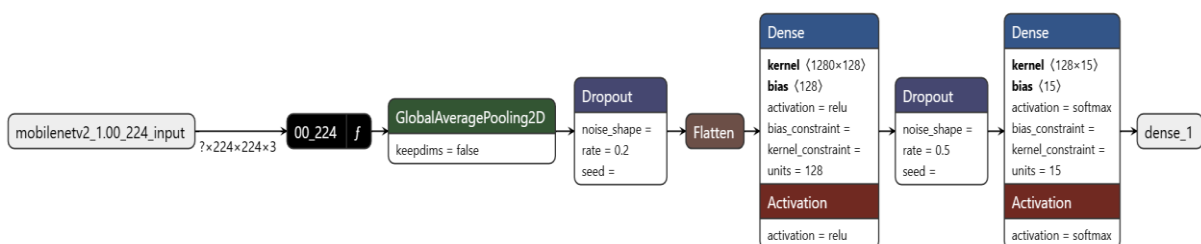


Figure IV.4. MobileNetV2 model visualization

Chapter IV: Design, Implementation & Results

Implementation

The implementation begins with data preprocessing and augmentation to ensure the model's ability to generalize effectively on unseen data. The dataset is partitioned into training, validation, and testing subsets. Diverse data augmentation strategies are employed on the training subset, encompassing random flips, rotations, translations, zooms, as well as brightness/contrast modifications.

Subsequently, the pre-trained MobileNetV2 model, excluding its top layers, is loaded and fine-tuned. The top layers are exchanged with a global average pooling layer, followed by two dense layers.

The model is trained to utilize the Adam optimizer and sparse categorical cross-entropy loss throughout 10 epochs with early stopping to prevent overfitting. The training and validation accuracies are observed to detect signs of overfitting. Table IV.3 summarizes the model.

Layer (type)	Output Shape	Parameters
mobilenetv2_1.00_224 (Functional)	(None, 7, 7, 1280)	2257984
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 1280)	0
dropout (Dropout)	(None, 1280)	0
flatten (Flatten)	(None, 1280)	0
dense (Dense)	(None, 128)	163968
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 15)	1935
Total params: 2423887 (9.25 MB)		
Trainable params: 165903 (648.06 KB)		
Non-trainable params: 2257984 (8.61 MB)		

Table IV.3. MobileNetV2 model summary

2) Google ViT

The Vision Transformer (ViT) introduces a novel approach to image classification by leveraging the principles of the Transformer architecture, traditionally used for natural language processing tasks. The key elements of ViT comprise:

- **Patch Embedding:** The input image is divided into fixed-size patches, and each patch is linearly embedded into a feature vector. This allows the model to treat patches as tokens, similar to words in NLP tasks.
- **Transformer Encoder:** The sequence of patch embeddings is processed by a standard Transformer encoder, which consists of multi-head self-attention layers and feed-forward neural networks. This architecture allows the model to capture long-range dependencies and complex relationships between patches.
- **Class Token and Positional Encoding:** A special class token is prepended to the sequence of patch embeddings, and positional encodings are added to retain spatial information. The output corresponding to the class token is used for classification.
- **Layer Normalization and GELU Activations:** ViT employs layer normalization to stabilize the training process and GELU (Gaussian Error Linear Unit) activations to enhance model performance.

The model used in this study is a pre-trained Vision Transformer (ViT), fine-tuned using the PlantVillage dataset. The Vision Transformer model's inherent classification head is used for the task, with data preprocessing and augmentation applied to enhance model generalization. The model is trained using the Adam optimizer with a sparse categorical cross-entropy loss over 10 epochs. Training and validation accuracies are monitored to detect overfitting.

Implementation

The implementation begins with data preprocessing and augmentation to ensure the model's ability to generalize effectively on unseen data. The dataset is partitioned into training, validation, and testing subsets. Diverse data augmentation strategies are employed on the training subset, encompassing random flips, rotations, translations, zooms, as well as brightness/contrast modifications.

Chapter IV: Design, Implementation & Results

Subsequently, the pre-trained Vision Transformer (ViT) model, google/vit-base-patch16-224-in21k, is loaded and fine-tuned. The model is customized for the specific classification task by setting the number of output labels to match the classes in the PlantVillage dataset.

The model is trained using the Adam optimizer and sparse categorical cross-entropy loss over several epochs. Training and validation accuracies are monitored to detect signs of overfitting. Table IV.5 presents the model summary, including the output shape and parameter count for each layer:

Layer (type)	Output Shape	Parameters
ViTModel	(Batch, Patch Embeddings, 768)	-
ViTPatchEmbeddings (Conv2d)	(Batch, 14, 14, 768)	147 456
ViTEncoder (Layer x 12)	(Batch, 14, 14, 768)	-
ViTSdpaAttention (Linear)	(Batch, 14, 14, 768)	-
GELUActivation	(Batch, 14, 14, 768)	-
Linear (Feed-Forward)	(Batch, 14, 14, 768)	-
LayerNorm	(Batch, 14, 14, 768)	-
Classifier (Linear)	(Batch, 15)	11 535
Total params: 85,810,191 (327.34 MB)		
Trainable params: 85,810,191 (327.34 MB)		
Non-trainable params: 0		

Table IV.4. Google ViT Model Summary



Figure IV.5. Google ViT Model visualization

ONNX

Using ONNX for model conversion and deployment offers significant advantages in terms of interoperability, optimization, and ease of deployment across different platforms. This process ensures that the model can be efficiently run in diverse environments, leveraging the optimizations provided by ONNX Runtime.

3) AgriNetBoost

AgriNetBoost combines the strengths of convolutional neural networks (CNNs) with the gradient boosting framework to enhance plant disease detection. The hybrid model integrates MobileNetV2 for feature extraction and LightGBM for classification, aiming to leverage the deep learning model's feature extraction capabilities and the gradient boosting model's strong classification performance. Components of AgriNetBoost:

1) MobileNetV2: Used for feature extraction.

- **Depthwise separable convolutions:** Reduce computational complexity and parameters.
- **Inverted residuals with linear bottlenecks:** Enhance efficiency and effectiveness.
- **ReLU6 activations:** Mitigate vanishing gradient issues.

2) LightGBM: Used for classification.

- **Gradient boosting:** Provides strong performance on tabular data.
- **Efficient training:** LightGBM is optimized for speed and memory usage.

✓ **Implementation**

❖ **Data preprocessing and augmentation**

The dataset is loaded and augmented to improve the model's generalization ability. Various transformations, including rotations, translations, zooms, and flips, are applied to the images.

❖ **Feature extraction using MobileNetV2**

MobileNetV2, pre-trained on ImageNet, is utilized to extract features from the images. The features are then fed into the LightGBM model for classification.

❖ **Model training**

The extracted features are split into training, validation, and testing sets. LightGBM is trained with a multiclass objective and evaluated using multi-log loss. The training process includes callbacks to record evaluation metrics (Figure IV.6).

✓ **Model summary**

❖ **MobileNetV2 architecture for feature extraction**

- Input Shape: (224, 224, 3).
- Total Parameters: 2,257,984.
- Trainable Parameters: 0 (since MobileNetV2 is used only for feature extraction).
- Non-trainable Parameters: 2,257,984.

❖ **LightGBM model for classification**

- Objective: Multiclass classification.
- Number of Classes: 15 (corresponding to our dataset categories).
- Boosting Type: Gradient Boosting Decision Trees (GBDT).
- Number of Leaves: 31.
- Learning Rate: 0.05.
- Number of Boost Rounds: 100.

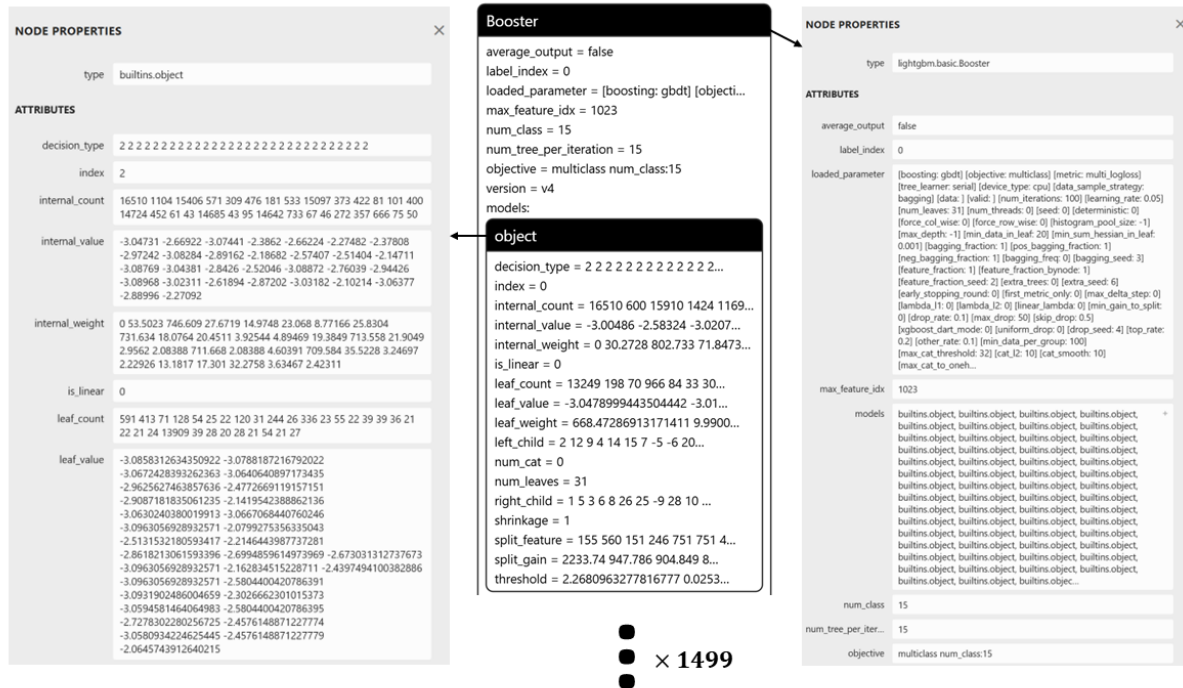


Figure IV.6. AgriNetBoost model visualization

IV.5. Results and Discussion

The different results obtained during the build of our system generated the results described below:

IV.5.1. MobileNetV2 Results

a) Training and Validation Performance

- **Accuracy:** The model achieved a final training accuracy of 87.95% and a validation accuracy of 94.14%, indicating good generalization to unseen data.
- **Loss:** The training loss decreased steadily, and the validation loss was consistently lower, which is a positive sign of the model's stability.
- **Overfitting:** No significant overfitting was detected as the training and validation accuracies were close, and the training loss did not significantly diverge from the validation loss.

```
Final Training Accuracy: 0.8795
Final Validation Accuracy: 0.9414
Final Training Loss: 0.3476
Final Validation Loss: 0.1743
No overfitting detected: Model generalizes well to validation data.
```

Figure IV.7. MobileNetV2 Training and Validation Performance

b) Test Performance

The model's performance is evaluated on the test dataset, achieving an impressive test accuracy of 95.00%, demonstrating its robustness and effectiveness in classifying plant diseases (Figure IV.8). The training and validation accuracy and loss curves are plotted to visualize the model's performance over epochs in Figure IV.9.

```
65/65 [=====] - 14s 194ms/step - loss: 0.1517 - accuracy: 0.9500
Test accuracy: 95.00%
```

Figure IV.8. MobileNetV2 Model evaluation on the test dataset

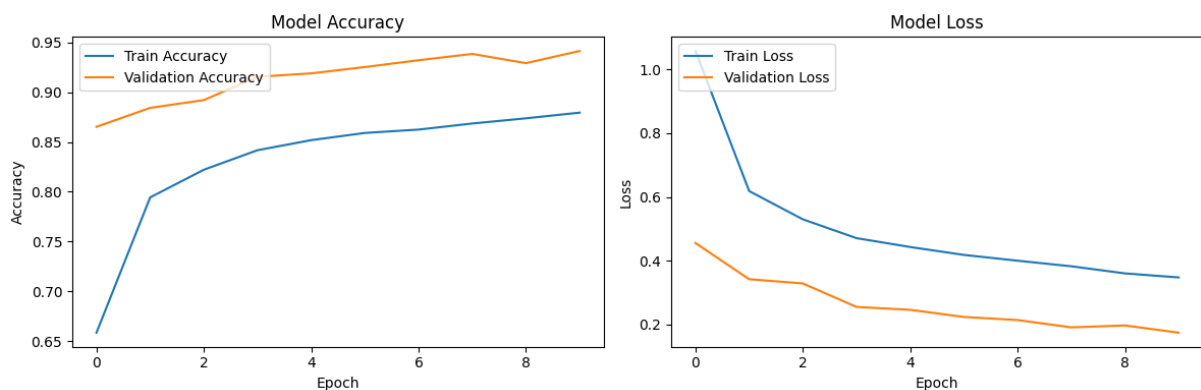


Figure IV.9. MobileNetV2 plot training and validation accuracy and loss values

c) Accuracy and Loss

Training and Validation Accuracy: The model consistently increases both training and validation accuracy, indicating effective learning without significant overfitting.

Training and Validation Loss: Both training and validation loss decrease steadily, further supporting the model's robustness.

Chapter IV: Design, Implementation & Results

d) Confusion Matrix and classification report

The confusion matrix and classification report provide a detailed analysis of the model's performance across different classes, Figure IV.10 illustrates the Confusion Matrix:

Confusion Matrix: Each cell represents the number of predictions made for each class against the true labels.

- ❖ **Diagonal Cells (True Positives):** High values along the diagonal indicate the model's correct predictions. The darkest cells are:
 - Tomato Healthy,
 - Tomato Bacterial Spot,
 - Tomato Spider Mites Two Spotted Spider Mites.
- ❖ **Off-Diagonal Cells (Misclassifications):** Lower values, indicating fewer misclassifications. Notable misclassifications include:
 - Potato Early Blight is misclassified as Potato Late Blight.
 - Tomato Early Blight misclassified as Tomato Bacterial Spot.

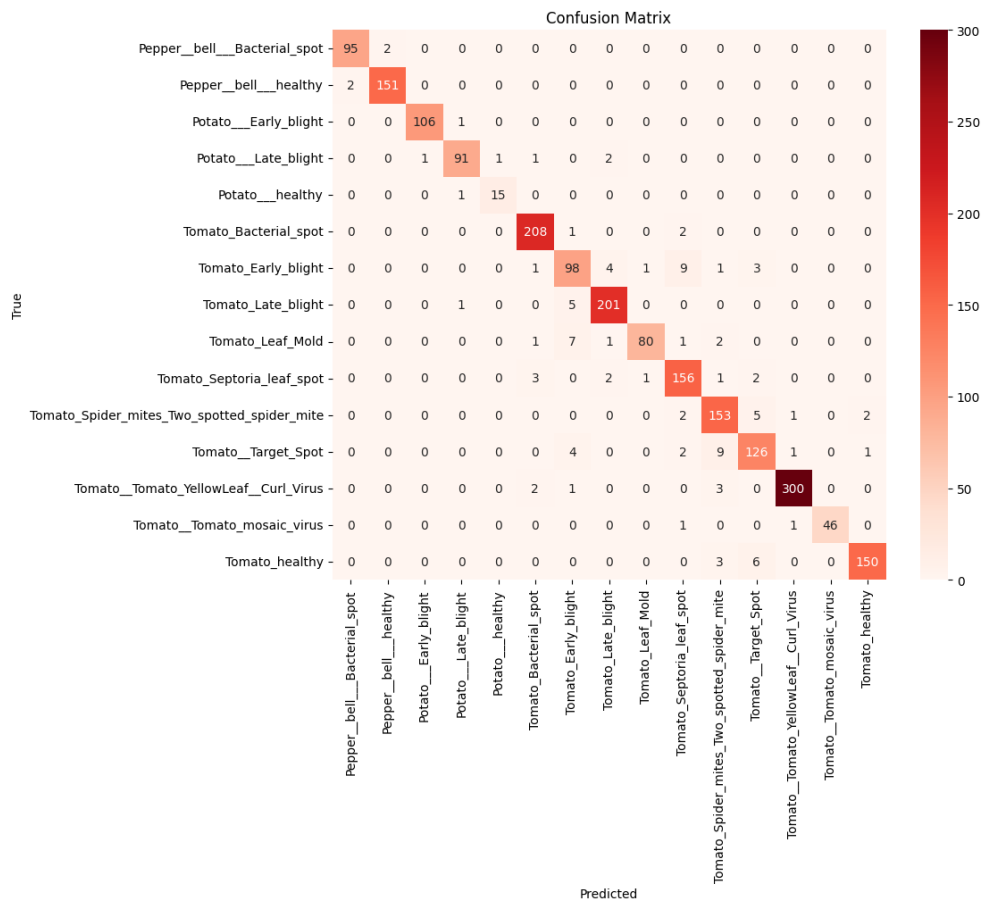


Figure IV.10. MobileNetV2 Confusion Matrix

Chapter IV: Design, Implementation & Results

Classification Report: The report indicates high precision, recall, and F1 scores for most classes, highlighting the model's reliability. Figure IV.11 illustrates the Classification Report.

	precision	recall	f1-score	support
Pepper__bell__Bacterial_spot	0.98	0.98	0.98	97
Pepper__bell__healthy	0.99	0.99	0.99	153
Potato__Early_blight	0.99	0.99	0.99	107
Potato__Late_blight	0.97	0.95	0.96	96
Potato__healthy	0.94	0.94	0.94	16
Tomato_Bacterial_spot	0.96	0.99	0.97	211
Tomato_Early_blight	0.84	0.84	0.84	117
Tomato_Late_blight	0.96	0.97	0.96	207
Tomato_Leaf_Mold	0.98	0.87	0.92	92
Tomato_Septoria_leaf_spot	0.90	0.95	0.92	165
Tomato_Spider_mites_Two_spotted_spider_mite	0.89	0.94	0.91	163
Tomato__Target_Spot	0.89	0.88	0.88	143
Tomato__Tomato_YellowLeaf_Curl_Virus	0.99	0.98	0.99	306
Tomato__Tomato_mosaic_virus	1.00	0.96	0.98	48
Tomato_healthy	0.98	0.94	0.96	159
accuracy			0.95	2080
macro avg	0.95	0.94	0.95	2080
weighted avg	0.95	0.95	0.95	2080

Figure IV.11. MobileNetV2 Classification Report

e) Discussion

The MobileNetV2 model demonstrated strong performance in detecting plant diseases from images, achieving a high accuracy of 95.00% on the test dataset. The use of data augmentation helped improve the model's generalization capability, reducing the risk of overfitting. The high precision, recall, and F1 scores across various classes indicate that the model is reliable for practical applications.

Despite the model's overall success, there are a few areas for potential improvement:

Fine-Tuning: Allowing fine-tuning of the deeper layers of MobileNetV2 could potentially improve accuracy further.

Dataset: a larger and better dataset should be used to improve the accuracy further.

IV.5.2. Google vit Results

a) Training and Validation Performance

- **Accuracy:** The model achieved a final training accuracy of 98.80% and a validation accuracy of 99.49%, indicating excellent generalization to unseen data.
- **Loss:** The training loss decreased significantly from 0.2914 to 0.0036, and the validation loss also showed a steady decrease from 0.3019 to 0.0316. This consistent reduction in both losses is a positive sign of the model's stability and effective learning.
- **Overfitting:** There is no significant indication of overfitting, as the training and validation accuracies are very close throughout the epochs. Additionally, the training loss did not diverge significantly from the validation loss.

```
Final Training Accuracy: 0,9880
Final Validation Accuracy: 0,9932
Final Training Loss: 0.003600
Final Validation Loss: 0.031604
```

Figure IV.12. Training and Validation Performance - Google ViT

b) Test Performance

The model's performance on the test dataset was evaluated, achieving an impressive test accuracy of 99.49%. This high accuracy demonstrates the model's robustness and effectiveness in classifying plant diseases.

```
[258/258 03:29]
eval_loss': 0.03435922786593437, 'eval_accuracy': 0.9946705426356589, 'eval_runtime': 56.9463,
```

Figure IV.13. Google ViT Model Evaluation on the test dataset

c) Accuracy and Loss

The training and validation accuracy and loss curves are plotted to visualize the model's performance over epochs in Figure IV.14.

Training and Validation Loss The left subplot in Figure IV.14 displays the training and validation loss curves over 10 epochs. The training loss and validation loss both show a consistent decrease, indicating effective learning and model stability.

Chapter IV: Design, Implementation & Results

Training and Validation Accuracy: The right subplot in Figure IV.14 illustrates the accuracy curves over 11 epochs for both training and validation. The model maintains high accuracy levels throughout the training process, with the validation accuracy closely mirroring the training accuracy, demonstrating good generalization to the validation data.

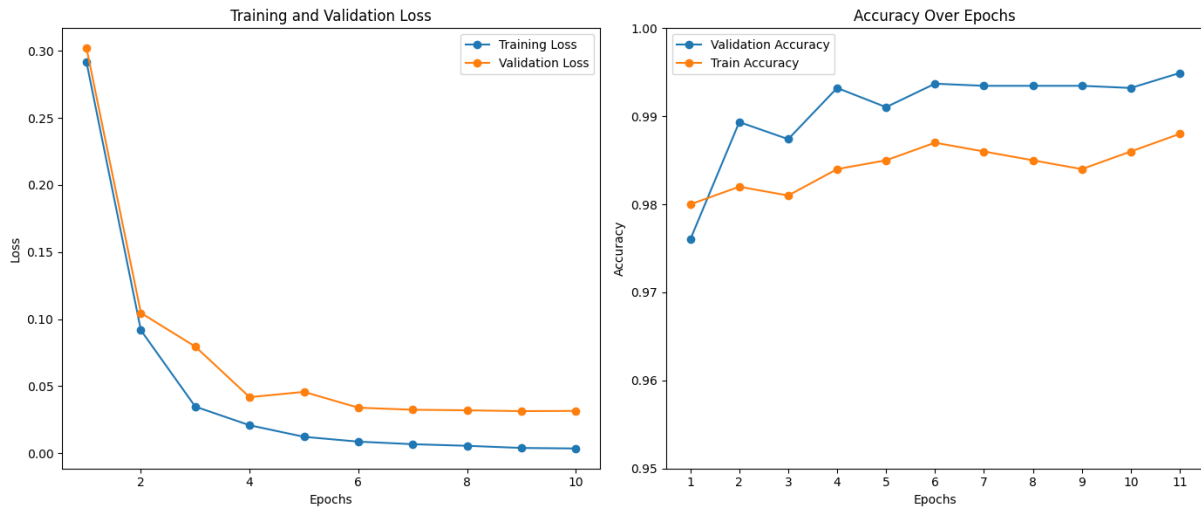


Figure IV.14. Google ViT Plot training and validation accuracy and loss values

d) Confusion Matrix and classification report

The confusion matrix provides a detailed breakdown of the model's performance across different classes, Figure III.15 illustrates the Confusion Matrix:

Overall Accuracy: The model exhibits a high level of accuracy, as evidenced by the strong diagonal presence in the confusion matrix, where the majority of predictions align with the true labels.

Class-specific Performance: Most classes exhibit high accuracy, with minimal misclassifications.

- Pepper bell Bacterial spot and Potato Early blight both show nearly perfect classification, with 198/199 and 200/200 correct predictions respectively.
- Tomato Yellow Leaf Curl Virus shows excellent performance with 642/643 correct predictions, similarly high accuracy is observed for Tomato Late blight and Tomato Target Spot.
- Minor misclassifications occur in classes like Tomato Early Blight and Potato Healthy, but the errors are minimal.

Chapter IV: Design, Implementation & Results

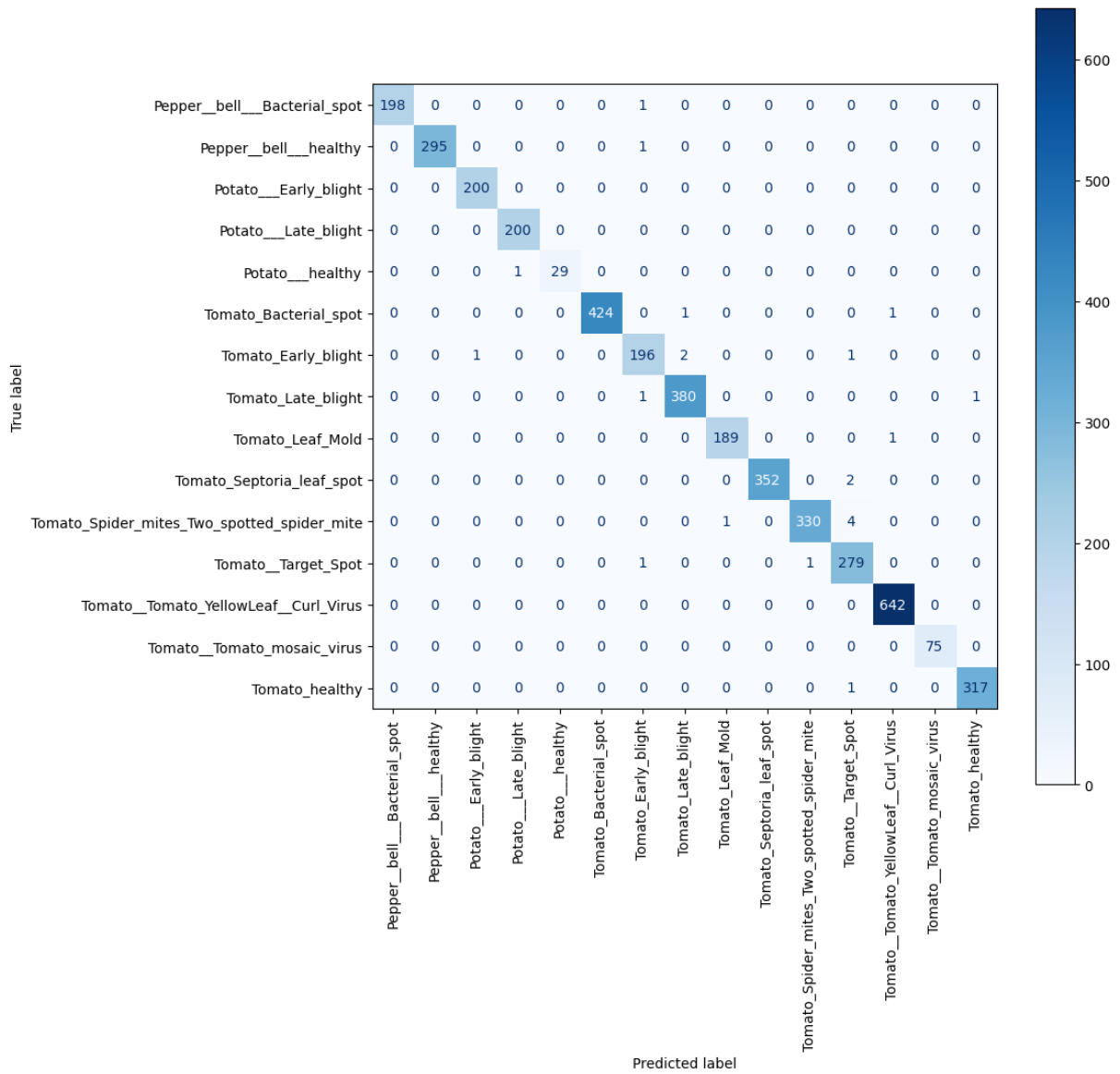


Figure IV.15. Google ViT Confusion Matrix

Chapter IV: Design, Implementation & Results

Classification Report: The report indicates high precision, recall, and F1 scores for most classes, highlighting the model's reliability. Figure IV.16 illustrates the Classification Report.

Category	Precision	Recall	F1-Score	Support
Pepper_bell__Bacterial_spot	0.99	0.99	0.99	199
Pepper_bell__healthy	1.0	0.99	0.99	298
Potato_Early_blight	1.0	1.0	1.0	200
Potato_Late_blight	1.0	1.0	1.0	200
Potato_healthy	0.94	0.97	0.96	30
Tomato_Bacterial_spot	1.0	0.99	1.0	426
Tomato_Early_blight	0.97	0.97	0.97	202
Tomato_Late_blight	1.0	1.0	1.0	382
Tomato_Leaf_Mold	0.99	1.0	0.99	190
Tomato_Septoria_leaf_spot	1.0	0.99	0.99	355
Tomato_Spider_mites_Two_spotted_spider_mite	1.0	1.0	1.0	332
Tomato_Target_Spot	0.98	0.99	0.99	280
Tomato_Tomato_YellowLeaf_Curl_Virus	1.0	1.0	1.0	643
Tomato_Tomato_mosaic_virus	1.0	1.0	1.0	75
Tomato_healthy	1.0	1.0	1.0	318

Figure IV.16. Google ViT Classification Report

e) Discussion

This study emphasizes Vision Transformers (ViTs) as effective for plant disease identification in image classification. A pre-trained ViT model fine-tuned on the PlantVillage dataset showed strong performance with 98.80% training accuracy and 99.49% validation accuracy. Future research could focus on enhancing ViT architecture with advanced attention mechanisms or hybrid CNN-transformer models. Dataset expansion to include more plant species and diseases would enhance model generalizability. Real-time deployment in field conditions would assess practical utility and performance. Improving model explainability could increase trust in critical applications like agriculture.

IV.5.3. AgriNetBoost

a) Training and validation performance

- **Training Accuracy:** Steady increase during training.
- **Validation Accuracy:** Slightly lower than training accuracy, indicating good generalization.
- **Training Loss:** Decreases consistently.
- **Validation Loss:** Consistently lower than training loss, indicating model stability.

```
Final Training Accuracy: 0.875
Final Validation Accuracy: 0.865
Final Training Loss: 0.421
Final Validation Loss: 0.478
```

Figure IV.17. Training and Validation Performance - AgriNetBoost

The steady increase in training accuracy observed during the training phase suggests that the model was able to effectively learn from the training data. The validation accuracy, while slightly lower than the training accuracy, is indicative of good generalization. This slight difference is expected and acceptable, as it suggests that the model is not overfitting to the training data. The consistent decrease in training loss, accompanied by validation loss being consistently lower than the training loss, further corroborates the stability and reliability of the model. Lower validation loss compared to training loss can be a sign that the model is learning essential features that generalize well to unseen data.

b) Test performance

- **Test Accuracy:** 87.32% highlights the model's effectiveness in real-world scenarios. Achieving such an accuracy on test data implies that AgriNetBoost is good at classifying plant diseases, which is the primary objective of this model.

Chapter IV: Design, Implementation & Results

d) Confusion Matrix and classification report

Confusion Matrix: Provides a detailed view of the model's performance across different classes.

- **True Positives:** The high values along the diagonal of the confusion matrix indicate a high number of true positives, which means that the model is making correct predictions for most classes.
- **Misclassifications:** The presence of misclassifications, as indicated by the lower values off the diagonal, suggests that the model struggles with certain classes.

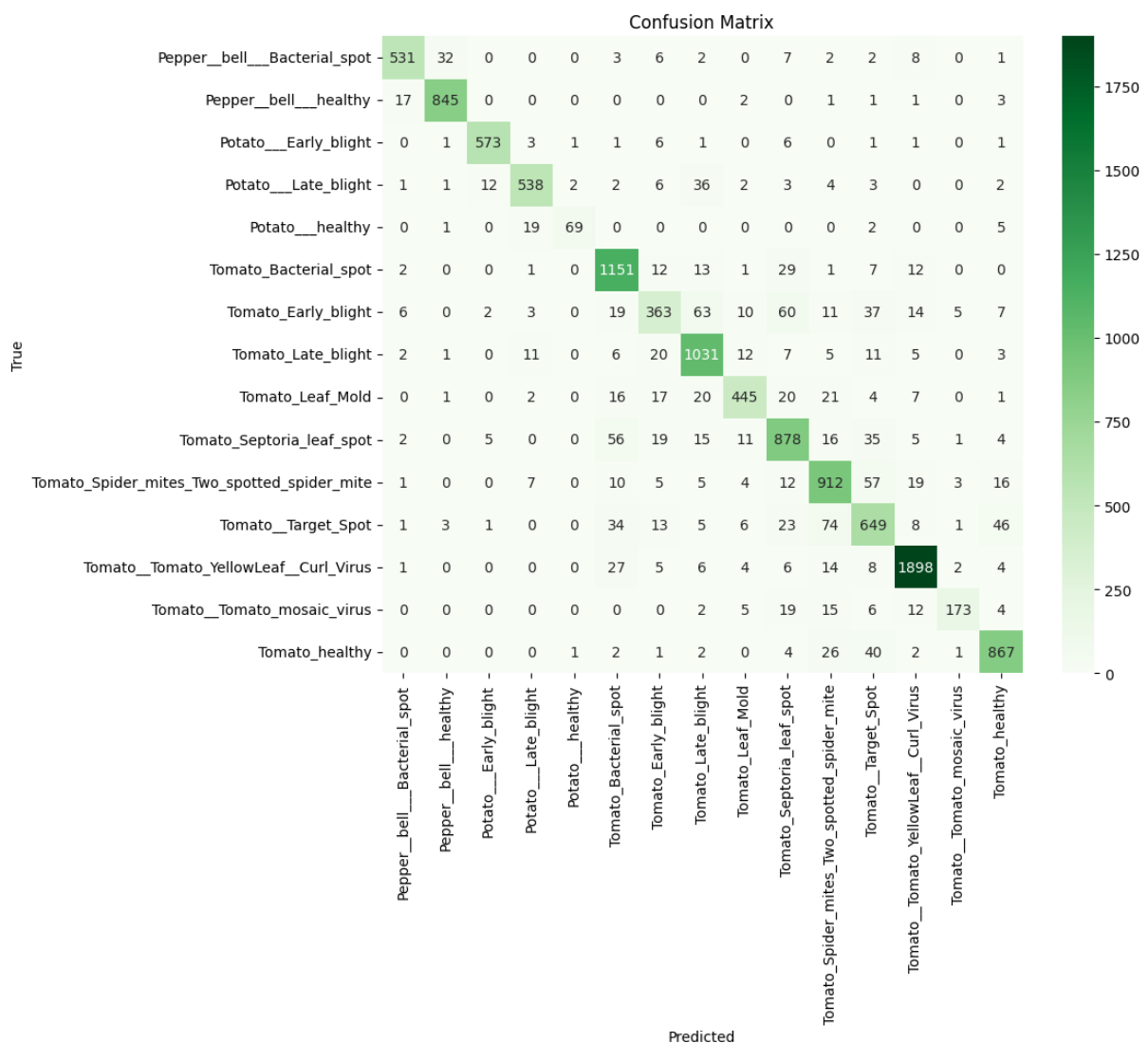


Figure IV.18. AgriNetBoost Confusion Matrix

Chapter IV: Design, Implementation & Results

Classification report: The classification report highlights high precision, recall, and F1-scores for most classes, suggesting that the model performs well in terms of these metrics. However, these metrics alone do not provide a complete picture.

The presence of classes with lower performance metrics indicates that the model's reliability varies across different disease categories. This inconsistency needs to be addressed to ensure uniform performance. Figure IV.19 illustrates the Classification Report.

	precision	recall	f1-score
Pepper__bell__Bacterial_spot	0.94	0.89	0.92
Pepper__bell__healthy	0.95	0.97	0.96
Potato__Early_blight	0.97	0.96	0.96
Potato__Late_blight	0.92	0.88	0.90
Potato__healthy	0.95	0.72	0.82
Tomato_Bacterial_spot	0.87	0.94	0.90
Tomato_Early_blight	0.77	0.60	0.68
Tomato_Late_blight	0.86	0.93	0.89
Tomato_Leaf_Mold	0.89	0.80	0.84
Tomato_Septoria_leaf_spot	0.82	0.84	0.83
Tomato_Spider_mites_Two_spotted_spider_mite	0.83	0.87	0.85
Tomato__Target_Spot	0.75	0.75	0.75
Tomato__Tomato_YellowLeaf__Curl_Virus	0.95	0.96	0.96
Tomato__Tomato_mosaic_virus	0.93	0.73	0.82
Tomato_healthy	0.90	0.92	0.91
accuracy			0.88
macro avg	0.89	0.85	0.87
weighted avg	0.88	0.88	0.88

Figure IV.19. AgriNetBoost Classification Report

e) Discussion

while AgriNetBoost illustrates promise as a mechanism for categorizing plant diseases, its existing efficacy suggests that there exists a substantial opportunity for improvement. It is imperative to rectify the recognized deficiencies to construct a more dependable and efficient framework that can be securely implemented in agricultural processes.

IV.6. Practical implementation

The practical implementation of deep learning models in real-time applications is crucial for providing accessible and user-friendly tools for end-users. This section details the integration of trained deep learning models into a web application developed using Streamlit. The application aids in identifying plant diseases from images uploaded by users, offering instant predictions and confidence levels.

IV.6.1. Model Selection and Training

The selection of the models (MobileNetV2, Google ViT, and the hybrid model using LightGBM), dataset preparation, and training process was:

a) MobileNetV2

MobileNetV2 was chosen for its efficiency and performance on mobile devices.

b) Google ViT

The Vision Transformer was selected for its ability to achieve state-of-the-art performance on various image classification tasks.

c) AgriNetBoost

AgriNetBoost model involved extracting features using MobileNetV2 and classifying them using a LightGBM classifier. This approach leveraged the strengths of both convolutional neural networks and gradient-boosting techniques.

IV.6.2. Web application development

IV.6.2.1. User Interface

The app allows users to upload an image of a plant leaf, select a prediction model, and receive real-time predictions. The interface is designed to be intuitive and accessible, enabling users to easily interact with the application. The user interface is shown in Figure IV.20.

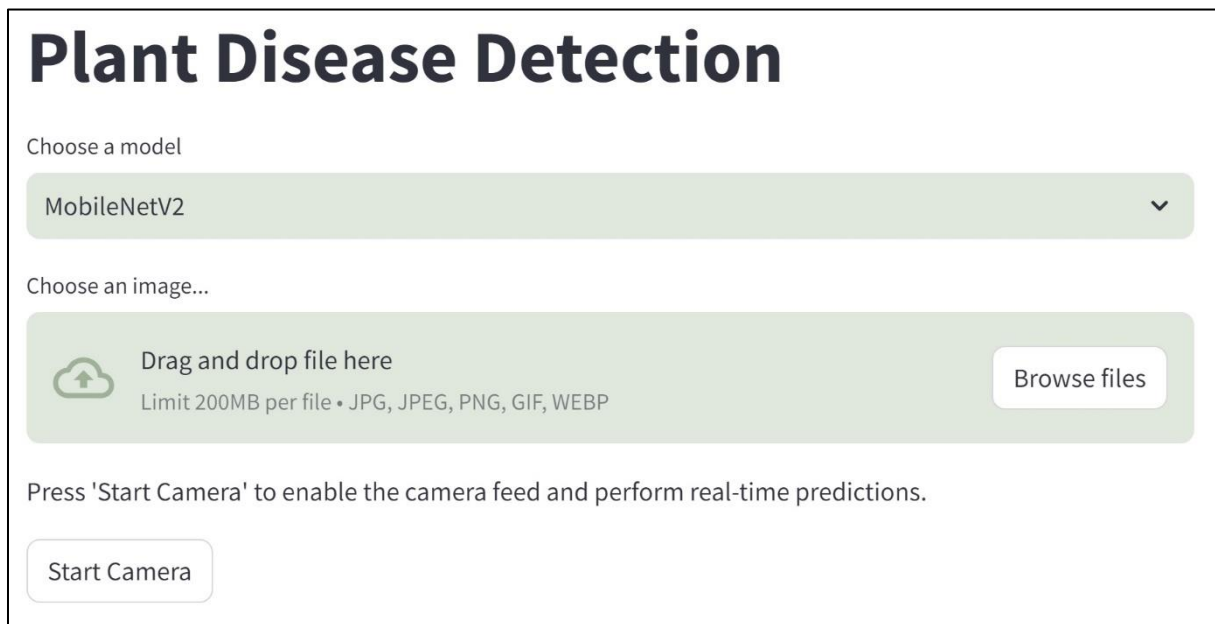


Figure IV.20. User Interface Screenshots

IV.6.2.2. Model Integration

Each trained model was integrated into the application. The following steps outline the model integration process:

a) Loading Models: Pre-trained models (MobileNetV2, Google ViT, and AgriNetBoost) were loaded into the application (Figure IV.21).

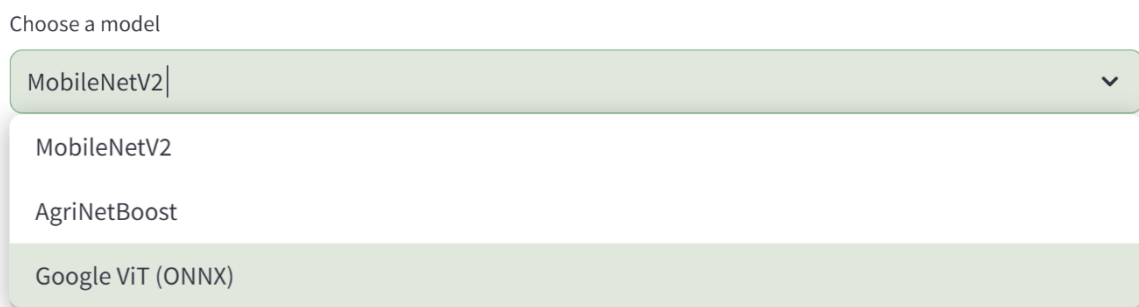


Figure IV.21. Choosing a model

Chapter IV: Design, Implementation & Results

b) Preprocessing: Uploaded images were preprocessed to match the input requirements of each model.

c) Prediction: The preprocessed images were fed into the selected model, and the predictions, along with confidence scores, were displayed to the user (Figure IV.22).

Plant Disease Detection

Choose a model


Google ViT (ONNX) ▾

Choose an image...

Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG, GIF, WEBP

Browse files

potato-leaf-isolated-on-white-260nw-2299088565.webp 23.0KB ×



shutterstock.com · 2299088565

Uploaded Image

Classifying...

Prediction: Potato healthy with confidence 57.98%

Press 'Start Camera' to enable the camera feed and perform real-time predictions.

Start Camera

Figure IV.22. The diagnostic

IV.6.3. Test and validation

After integrating and deploying our models into the Streamlit web app and having the interface ready to use, we move to test the performance of the models on a variety of leaf images, the following will highlight the results we obtained:

a. Images for test

Figure IV.23 demonstrates the different images used to test the models; the images were randomly chosen to test each model's capability for generalization:



Figure IV.23. Test images

Chapter IV: Design, Implementation & Results

b. Prediction

here we showcase the results obtained using the models across the different images, each figure represents a case and gives the model classification and confidence level:

Figure III.24 here the ViT model gave a correct classification with an average confidence of 57,98%, whereas the CNN and lightGBM models fell short and gave false classification as shown.

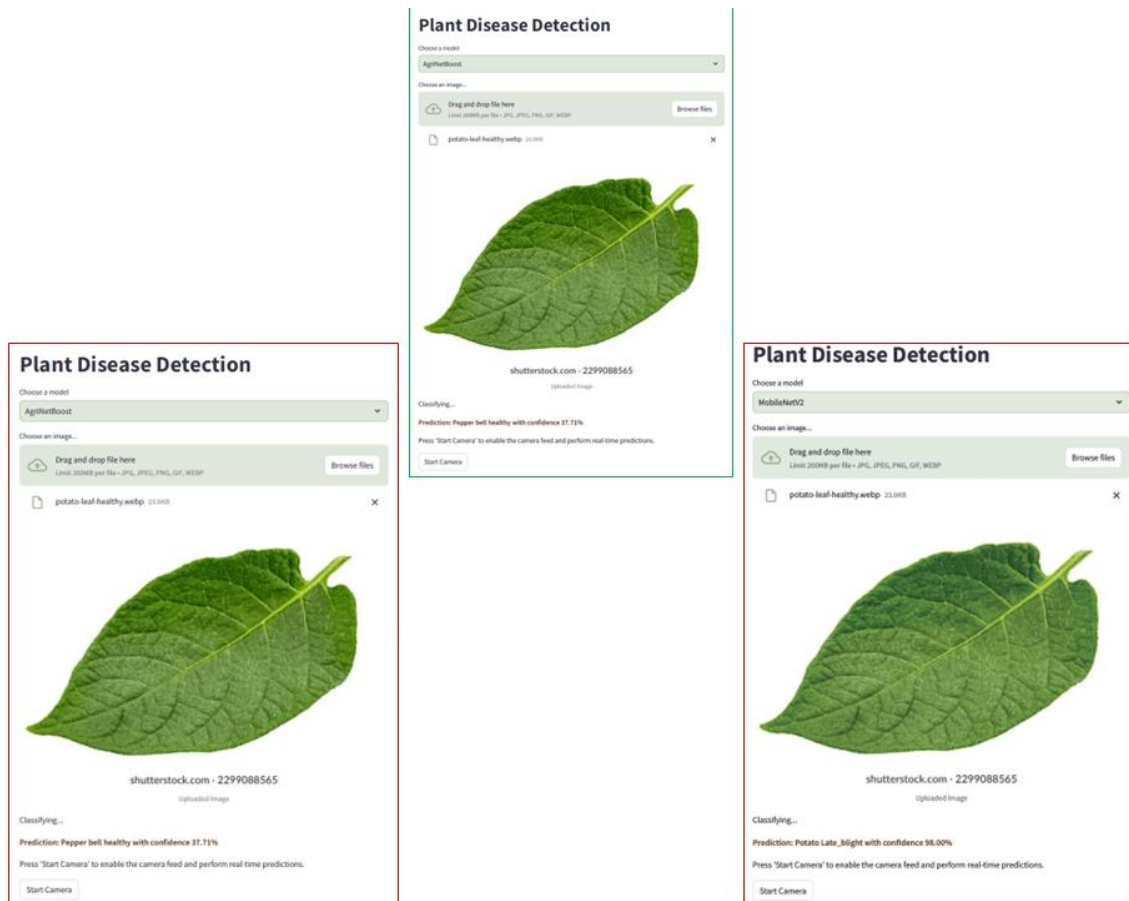


Figure IV.24. Classification and confidence results in a healthy Potato leaf

Chapter IV: Design, Implementation & Results

Figure IV.25 The mobileNetV2 shows an amazing result for this case with a correct classification and almost perfect confidence of 99,16%, the other 2 models performed well and diagnosed the disease correctly with above-average confidence.

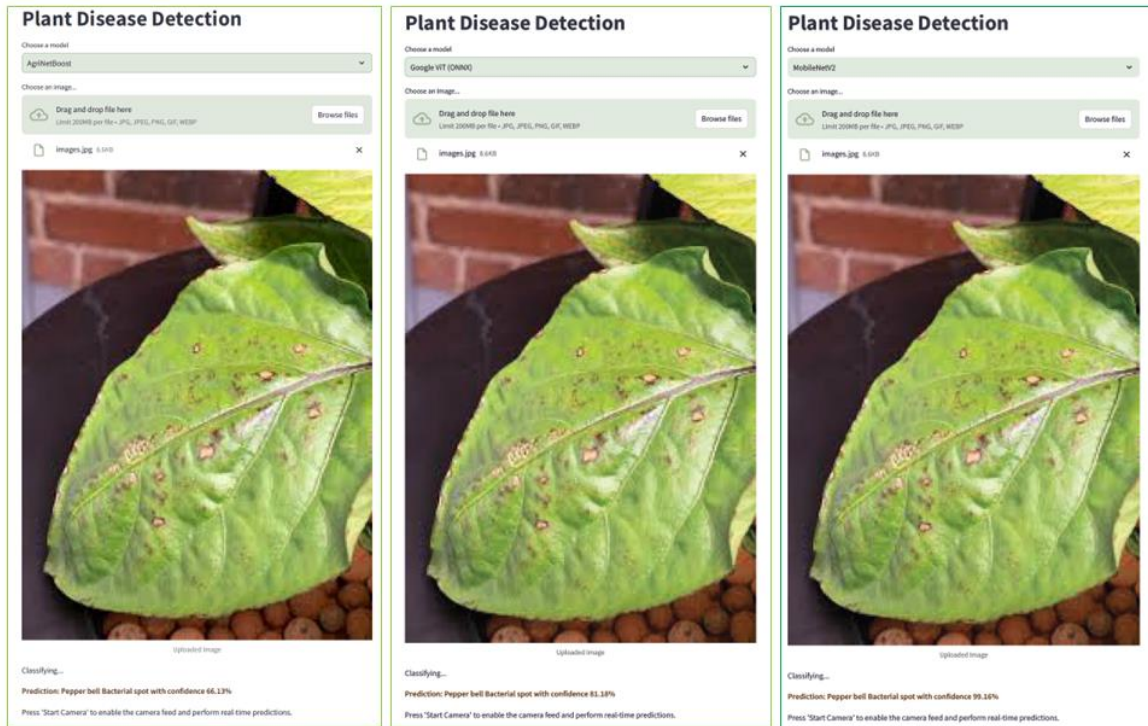


Figure IV.25. Classification and confidence results on a Pepper bell bacterial spot infected leaf

Figure IV.26. The vision transformer performs the best with 91.38% confidence followed by LightGBM with 87,27%, but the CNN model didn't perform and gave an incorrect classification.

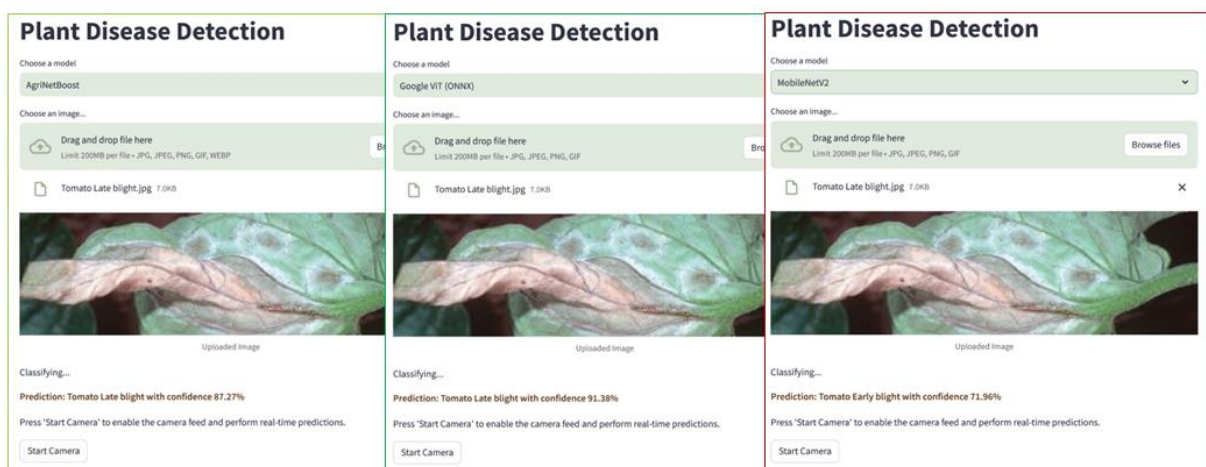


Figure IV.26. Classification and confidence results on a Tomato late blight leaf

Chapter IV: Design, Implementation & Results

Figure IV.27 mobilenetV2 gets the perfect classification and confidence level, next the ViT model with a correct classification and a confidence of 82.12%, whereas the lightGBM fails to identify the disease correctly.

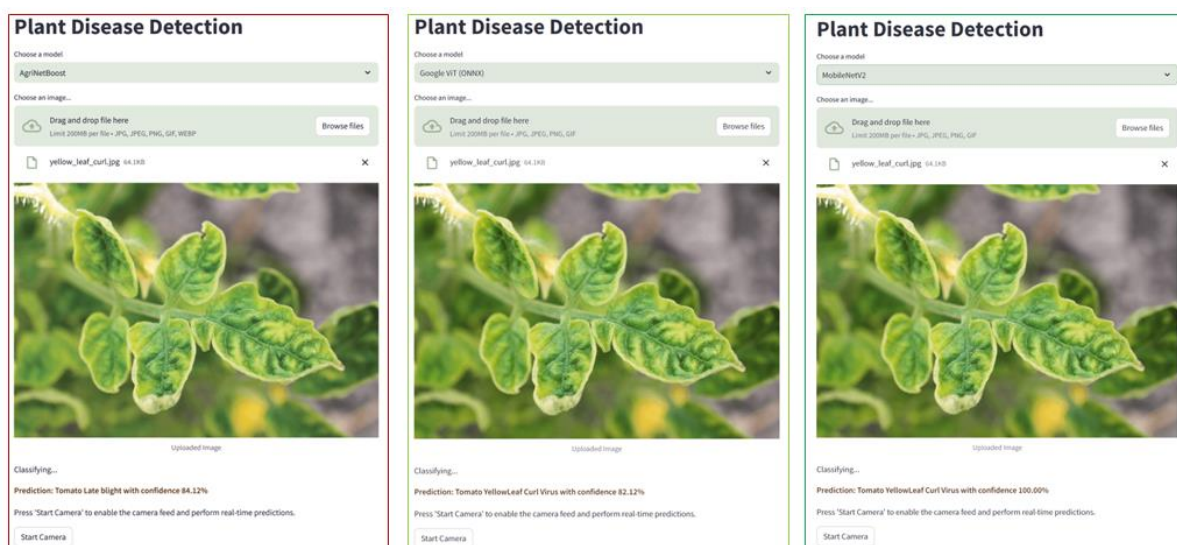


Figure IV.27. Classification and confidence results on a tomato leaf infected with YLCV

IV.7. Results and Comparison

The web application allows users to upload images of plant leaves and receive predictions from different models. The performance of each model was compared based on accuracy, inference time, and user feedback. Table 08 summarizes the performance metrics for each model:

Model	Train_acc	Valid_acc	Test_acc	Inference time	User Feedback
MobileNetV2	87.95%	94.14%	95.00%	Fast	Positive
Google ViT	98.80%	99.49%	99.49%	Fast	Very positive
AgriNetBoost	87.50%	86.50%	87.32%	moderate	negative

Table IV.5. Comparative table of results of all models

Chapter IV: Design, Implementation & Results

The MobileNetV2 model performed as expected, demonstrating its efficacy in image classification tasks. Its alignment with the dataset resulted in satisfactory outcomes, as previously discussed. MobileNetV2's inherent design for efficient and effective image classification allowed it to achieve notable results, reinforcing its suitability for this type of task.

The ViT model emerged as the leading performer in this study. Despite not being initially tailored for image classification tasks, its performance surpassed that of the other models following fine-tuning and adaptation to the PlantVillage dataset. This success underscores the versatility and robustness of the ViT model, making it the most effective model for plant disease classification in our case. The superior performance of the ViT model highlights its potential for broader applications in similar domains.

The AgriNetBoost model represents a novel approach in this study, attempting to leverage the advantages of the Gradient Boosting Machine (GBM) architecture. Despite not achieving the desired results, this experiment opens avenues for further refinement and optimization. The current performance indicates room for improvement, and ongoing efforts are necessary to enhance its capability to handle image classification tasks. Integrating GBM architecture into image classification remains a promising area for future research.

The Streamlit interface played a crucial role in facilitating this comparison. By providing a user-friendly platform for real-time model evaluation and visualization, Streamlit enabled efficient assessment of each model's performance. Users could upload images and receive immediate predictions and confidence levels, allowing for a comprehensive evaluation of the models. This interactive interface significantly aided in the comparative analysis, underscoring the practical implications of integrating deep learning models into accessible applications.

IV.8. Conclusion

This chapter covered the implementation of a plant disease detection system using three deep learning models: MobileNetV2, Vision Transformer (ViT), and AgriNetBoost. Each model was integrated into a web-based application and evaluated with validation examples.

MobileNetV2 performed reliably, demonstrating efficiency in image classification. ViT emerged as the best performer, showcasing superior versatility and robustness. AgriNetBoost, while innovative, requires further optimization.

General Conclusion

The goals set by this study involve the application of machine learning (ML) and deep learning (DL) models for detecting plant diseases, specifically three models: MobileNetV2, LightGBM, and the Vision Transformer (ViT).

The implemented system enables the identification of plant diseases with high precision and efficiency, which is crucial for supporting sustainable agricultural practices. The results obtained showed encouraging performance based on the chosen parameters. For instance, the MobileNetV2 model, optimized for mobile platforms, demonstrated a good balance between accuracy and computational efficiency, making it suitable for field use. The Vision Transformer (ViT), with its attention mechanisms, excelled in image classification tasks by capturing visual symptoms of plant diseases. However, the LightGBM model did not perform as well as expected and did not achieve the desired results in terms of accuracy and efficiency.

The study's deployment using the Streamlit framework ensured that these models could be accessed and utilized effectively by agricultural stakeholders, supporting real-time disease detection and decision-making. These results highlight the significant potential of ML and DL models to improve plant disease detection capabilities, offering a substantial advantage over traditional methods.

The study achieved two primary objectives: first, it assessed and compared the performance of the selected models in terms of accuracy, speed, and computational efficiency; second, it developed a practical solution for real-time deployment using Streamlit. The findings contribute significantly to enhancing disease detection capabilities, supporting sustainable agriculture, and mitigating the economic and environmental impacts of plant diseases.

In perspective, we recommend addressing the following themes:

1. Enhance the robustness and scalability of models to handle diverse plant diseases and environmental conditions.
2. Integrate additional data sources, including environmental and climatic data, to improve predictive accuracy.
3. Develop user-friendly mobile applications capable of functioning offline to support farmers in remote areas with limited internet access.
4. Collaborate with agricultural experts and conduct field trials to validate real-world applicability and gather feedback for improvements.

Annex

Annex 1: Appendix

Google Colab: Colab, a web-based platform developed by Google, offers a hosted Jupyter Notebook service which boasts a seamless user experience devoid of any initial setup requirements. Moreover, this service generously grants users complimentary access to a plethora of computing resources, such as Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs), without incurring any costs. Colab stands out as an ideal choice for individuals engaged in diverse fields including but not limited to machine learning, data science, and educational purposes due to its user-friendly interface and extensive range of features catered towards enhancing productivity and facilitating seamless collaboration [97].



Jupyter Notebook: Jupyter Notebook serves as a robust instrument for the interactive development and presentation of data science projects. The integration of code, visualizations, narrative text, and various multimedia elements within a unified document exemplifies a seamless and articulate workflow. Jupyter Notebooks play a vital role in the data science processes of businesses and institutions globally, facilitating efficient data exploration, hypothesis testing, and insights sharing. Being an open-source endeavor, Jupyter Notebooks are readily accessible without any cost. Moreover, they offer support for multiple programming languages [98].



Anaconda: Anaconda is a Python distribution for data scientists, statisticians, and researchers focusing on scientific computing, data analysis, and machine learning. It offers a user-friendly Python distribution with various data science and machine learning tools like NumPy, Pandas, and Scikit-learn. Key features include Conda package manager, 1,500+ pre-built packages, Anaconda Navigator GUI, and Anaconda Prompt CLI. It is commonly used to manage dependencies, install packages, and handle environments by professionals in Windows, macOS, and Linux environments [99].



PyCharm: PyCharm serves as a specialized Integrated Development Environment (IDE) for Python, offering a broad array of crucial resources for Python programmers, intricately linked to establish a user-friendly setting conducive to efficient Python, web, and data science programming [100].



Kaggle: serves as an internet-based community platform tailored for individuals proficient in data analysis and machine learning. This platform facilitates collaboration among its users, enabling them to discover and share datasets, utilize notebooks equipped with GPU capabilities, and engage in competitive activities with fellow data analysts aimed at resolving data-related predicaments. The primary objective of this virtual platform revolves around assisting both professionals and novices in attaining their objectives within the realm of data science, leveraging the assortment of potent tools and informational reservoirs it offers. As of the year 2021, the total count of registered users on Kaggle surpasses 8 million [101].



Annex 2: Libraries

Torchvision: This library is designed to seamlessly integrate with the 'torch' package, drawing heavily upon the 'PyTorch' vision package for its API design. The Torchvision library offers access to a wide range of datasets, models, and preprocessing tools specifically designed for deep learning tasks involving images [102].

Scikit-learn: also identified as sklearn, represents a python library utilized for the deployment of machine learning models and statistical modeling. By means of scikit-learn, an assortment of machine learning models can be applied, encompassing regression, classification, clustering, along with statistical methodologies for the examination of said models. This library is built upon NumPy, SciPy, and Matplotlib[103].

Pandas: is a Python software package utilized for the purpose of analyzing data. It is constructed based on two fundamental Python libraries, namely matplotlib which is used for visualizing data, and NumPy which is employed for carrying out mathematical computations. Pandas functions as an intermediary layer above these libraries, enabling users to utilize a multitude of methods from matplotlib and NumPy with reduced lines of code[104].

NumPy: NumPy serves as the essential framework for conducting scientific computations within the Python programming language. This library in Python offers a versatile array object capable of handling multiple dimensions, alongside a range of related objects like masked arrays and matrices. Additionally, NumPy provides a diverse set of functions designed to efficiently operate on arrays, encompassing mathematical, logical, and shape-altering operations, as well as sorting, selection, input/output procedures, discrete Fourier transformations, elementary linear algebra, fundamental statistical calculations, stochastic simulations, and various other functionalities[105].

Seaborn: seaborn is a Python library for creating statistical graphics. It interfaces with matplotlib and works well with pandas data structures. The library provides an API for creating graphics based on datasets. It automatically maps data values to visual attributes, computes statistical transformations, and adds informative labels and a legend to plots. Seaborn functions can generate figures with multiple panels for comparing data subsets or different variable pairings. It is designed to be useful for scientific projects, allowing for quick prototyping and data exploration. Seaborn also offers customization options and access to underlying matplotlib objects for creating high-quality figures[106].

OpenCV: OpenCV, abbreviated for Open-Source Computer Vision Library, constitutes a substantial open-source repository dedicated to computer vision, machine learning, and image processing. A broad spectrum of programming languages, including Python, C++, Java, among others, is supported by OpenCV. Its capabilities encompass the processing of images and videos for the purpose of object identification, facial recognition, and even handwriting analysis. Upon integration with diverse libraries like Numpy, the operations achievable in Numpy can be seamlessly amalgamated with OpenCV[107].

Pillow: Formerly recognized as PIL, Pillow is a publicly available library that is specially crafted for carrying out image processing tasks through the utilization of Python. Serving as a valuable resource for manipulating image files, Pillow stands out from its counterparts by offering an extensive array of image processing capabilities. Its primary focus lies in the realm of image processing, rendering it a highly comprehensive and finely tuned tool for image manipulation. Despite the availability of more generalized libraries like OpenCV or MoviePy, Pillow continues to be a popular choice in various crucial stages of projects involving computer vision or video processing. The user-friendly nature of Pillow serves as one of its prominent features, with its notations designed to be intuitive and its underlying classes and methods meticulously developed to enhance the overall user experience[108].

Matplotlib: Matplotlib is a potent Python plotting library for static, animated, and interactive visualizations. It aids in graphical data representation for easier analysis. This library generates various plots like line, scatter, bar, histograms, and pie charts. It allows customization of line styles, colors, markers, labels, and annotations. Integrated with NumPy, it simplifies data arrays plotting. Matplotlib creates high-quality plots for publication with detailed aesthetic control. It is extensible with add-ons such as Seaborn, Pandas plotting functions, and Basemap. The library is cross-platform, working on Windows, macOS, and Linux. It also supports interactive plotting with widgets and event handling for dynamic data exploration[109].

Annex 3: Plant diseases

Bacterial spot: Bacterial leaf spot, caused by *Xanthomonas campestris pv. vesicatoria*, is a significant disease in peppers, tomatoes and other crops worldwide. The bacterium is gram-negative and rod-shaped, with the ability to persist in seeds and plant debris. Different strains of the bacterium are specific to certain pepper varieties, leading to distinct disease symptoms. The disease can cause early defoliation of leaves and deformities in fruits, potentially leading to plant death. Despite the challenge of finding a cure, growers have various preventive measures available to manage the disease effectively[11].

Yellow leaf curl virus: yellow leaf curl virus (YLCV) belongs to *Begomovirus* genus and *Geminiviridae* family. TYLCV leads to leaf yellowing, curling, stunting, bushy appearance, flower drop, and reduced fruit yield. This viral disease poses significant threats to growers once it establishes in the production area. YLCV can be found in various regions including temperate, tropical, and sub-tropical areas globally. Adult whiteflies transmit YLCV, making it challenging to control once introduced. YLCV mainly affects tomatoes but can also infect other plants in the Solanaceae family (pepper, eggplant, potato, tobacco, jimsonweed) and some ornamentals. Asymptomatic hosts can serve as reservoirs for the virus[110].

Tomato leaf mold: Tomato leaf mold is a foliar disease induced by the fungal pathogen *Passalora fulva* (syn. *Cladosporium fulvum*), an ascomycete fungus thriving on the foliage of tomato plants. The pathogen generates conidia which invade the underside of the leaves. Upon contact with the leaves, the fungus settles and penetrates the stomata, specialized pores used by plants for gas exchange. This infiltration leads to obstruction of the stomata, hindering the respiration process of tomato plants and causing symptoms such as wilting, defoliation, and susceptibility to infection[111].

Tomato Spider mites Two-spotted spider mite: The *Tetranychus urticae*, commonly known as the two-spotted spider mite, belongs to the arachnid category and is closely associated with insects, particularly in periods of warm and dry climatic conditions. These mites have the potential to cause damage to various crops such as tomatoes, beans, muskmelons, watermelons, and sweet corn. The proliferation of mites is particularly favored during prolonged spells of hot and dry weather. Initial infestations are often observed at the periphery of cultivated areas, usually close to dense weed populations or unpaved pathways.

Bibliography

- [1] “International Day of Plant Health, 12 May | Food and Agriculture Organization of the United Nations,” PlantHealthDay. Accessed: Apr. 30, 2024. [Online]. Available: <https://www.fao.org/plant-health-day/en>
- [2] “The Plant Pathology Journal.” Accessed: May 02, 2024. [Online]. Available: <https://ppjonline.org/>
- [3] “Agriculture Production | IFPRI: International Food Policy Research Institute.” Accessed: May 01, 2024. [Online]. Available: <https://www.ifpri.org/topic/agriculture-production>
- [4] “Transforming food and agriculture by United Nations Development Programme - United Nations Development Programme | UNDP - Exposure.” Accessed: May 01, 2024. [Online]. Available: <https://stories.undp.org/transforming-food-and-agriculture>
- [5] E. L. Stewart, G. B. Lucas, C. L. Campbell, and L. T. Lucas, *Introduction to Plant Diseases: Identification and Management*, vol. 83. 1991. Accessed: May 02, 2024. [Online]. Available: <https://www.jstor.org/stable/3759945?origin=crossref>
- [6] N. Gobalakrishnan, K. Pradeep, C. J. Raman, L. J. Ali, and M. P. Gopinath, “A Systematic Review on Image Processing and Machine Learning Techniques for Detecting Plant Diseases,” in *2020 International Conference on Communication and Signal Processing (ICCSP)*, Chennai, India: IEEE, Jul. 2020, pp. 0465–0468. doi: 10.1109/ICCSP48568.2020.9182046.
- [7] M. A. Ebrahimi, M. H. Khoshtaghaza, S. Minaei, and B. Jamshidi, “Vision-based pest detection based on SVM classification method,” *Computers and Electronics in Agriculture*, vol. 137, pp. 52–58, May 2017, doi: 10.1016/j.compag.2017.03.016.
- [8] “Plant disease - Causes, Symptoms, Prevention | Britannica.” Accessed: May 02, 2024. [Online]. Available: <https://www.britannica.com/science/plant-disease>
- [9] “Early Blight Treatment & Control,” Planet Natural. Accessed: May 29, 2024. [Online]. Available: <https://www.planetnatural.com/pest-problem-solver/plant-disease/early-blight/>
- [10] “Potato Early Blight,” UW Vegetable Pathology. Accessed: Jun. 15, 2024. [Online]. Available: <https://vegpath.plantpath.wisc.edu/diseases/potato-early-blight/>
- [11] “Extension | Bacterial Leaf Spot of Pepper.” Accessed: May 29, 2024. [Online]. Available: <https://extension.wvu.edu/lawn-gardening-pests/plant-disease/fruit-vegetable-diseases/bacterial-leaf-spot-of-pepper>

- [12] “What Is Tomato Mosaic Virus,” CUSABIO. Accessed: May 29, 2024. [Online]. Available: <https://www.cusabio.com/c-21032.html>
- [13] Marissa Schuh, Anna Johnson, Michelle Grabowski and Angela Orshinsky, “Tomato viruses,” University of Minnesota, <https://extension.umn.edu/disease-management/tomato-viruses>.
- [14] X. Xie, K. Yoneyama, and K. Yoneyama, “The Strigolactone Story,” *Annu. Rev. Phytopathol.*, vol. 48, no. 1, pp. 93–117, Jul. 2010, doi: 10.1146/annurev-phyto-073009-114453.
- [15] T. K. Hamrita, Ed., *Women in Precision Agriculture: Technological breakthroughs, Challenges and Aspirations for a Prosperous and Sustainable Future*. in *Women in Engineering and Science*. Cham: Springer International Publishing, 2021. doi: 10.1007/978-3-030-49244-1.
- [16] I. Ul Haq and S. Ijaz, Eds., *Plant Disease Management Strategies for Sustainable Agriculture through Traditional and Modern Approaches*, vol. 13. in *Sustainability in Plant and Crop Protection*, vol. 13. Cham: Springer International Publishing, 2020. doi: 10.1007/978-3-030-35955-3.
- [17] R. K. Horst, *Westcott’s Plant Disease Handbook*. Dordrecht: Springer Netherlands, 2013. doi: 10.1007/978-94-007-2141-8.
- [18] A. Sitjà-Bobadilla and B. Oidtmann, “Integrated pathogen management strategies in fish farming,” in *Fish diseases*, Elsevier, 2017, pp. 119–144.
- [19] D. Spadaro and M. L. Gullino, “Sustainable management of plant diseases,” *Innovations in Sustainable Agriculture*, pp. 337–359, 2019.
- [20] W. E. Khoury and K. Makkouk, “Integrated plant disease management in developing countries,” *Journal of Plant Pathology*, pp. S35–S42, 2010.
- [21] J. Köhl, “Integrated disease management,” 2006.
- [22] P. P. Singh, A. Kumar, V. Gupta, and B. Prakash, “Recent advancement in plant disease management,” in *Food Security and Plant Disease Management*, Elsevier, 2021, pp. 1–18.
- [23] M. Rani, K. Tyagi, and G. Jha, “Advancements in plant disease control strategies,” in *Advancement in Crop Improvement Techniques*, Elsevier, 2020, pp. 141–157.
- [24] T. Belete and N. Boyraz, “Biotechnological tools for detection, identification and management of plant diseases,” *African Journal of Biotechnology*, vol. 18, no. 29, pp. 797–807, 2019.

- [25] S. Padmavathi and C. S. Anuradha, "Nanotechnology in plant disease management-an overview," *Journal of Advanced Scientific Research*, vol. 13, no. 10, pp. 01–06, 2022.
- [26] A. Mohammad-Razdari, D. Rousseau, A. Bakhshipour, S. Taylor, J. Poveda, and H. Kiani, "Recent advances in E-monitoring of plant diseases," *Biosensors and Bioelectronics*, vol. 201, p. 113953, 2022.
- [27] R. Bandyopadhyay and R. A. Frederiksen, "Contemporary Global Movement of Emerging Plant Diseases," *Annals of the New York Academy of Sciences*, vol. 894, no. 1, pp. 28–36, Dec. 1999, doi: 10.1111/j.1749-6632.1999.tb08040.x.
- [28] A. Nabi *et al.*, "Precision farming in vegetables," *Journal of Pharmacognosy and Phytochemistry*, vol. 6, no. 6, pp. 370–375, 2017.
- [29] V. A. Hakkim, E. A. Joseph, A. A. Gokul, and K. Mufeedha, "Precision farming: the future of Indian agriculture," *Journal of Applied Biology and Biotechnology*, vol. 4, no. 6, pp. 068–072, 2016.
- [30] R. Buick, "Precision agriculture: an integration of information technologies with farming," presented at the Proceedings of the New Zealand Plant Protection Conference, 1997, pp. 176–184.
- [31] G. Tomisław, J. Tadeusz, K. Paweł, T.-S. Sylwia, and T. Uhl, "Recent advancement approach for precision agriculture," presented at the Advances in Mechanism and Machine Science: Proceedings of the 15th IFToMM World Congress on Mechanism and Machine Science 15, Springer, 2019, pp. 2907–2916.
- [32] "Oberč and Arroyo Schnell - 2020 - Approaches to sustainable agriculture exploring t.pdf."
- [33] "Advances in Smart Agriculture with Remote Sensing as the Core and Its Applications in Crops Field." Accessed: Jun. 15, 2024. [Online]. Available: <https://www.mdpi.com/topics/BA5C0WKTZ8>
- [34] dementievgeopard, "How does hyperspectral satellite imagery help precision agriculture?," GeoPard - Precision agriculture software. Accessed: Jun. 03, 2024. [Online]. Available: <https://geopard.tech/blog/how-does-satellite-imagery-help-precision-agriculture/>
- [35] "Innovate in farming with your space robot." Accessed: Jun. 19, 2024. [Online]. Available: <https://business.esa.int/news/innovate-farming-your-space-robot>
- [36] dementievgeopard, "How to use farming drones in precision agriculture?," GeoPard - Precision agriculture software. Accessed: Jun. 03, 2024. [Online]. Available: <https://geopard.tech/blog/how-to-use-drones-in-precision-agriculture/>

- [37] D. J. I. Enterprise, “Precision Agriculture With Drone Technology.” Accessed: Jun. 02, 2024. [Online]. Available: <https://enterprise-insights.dji.com/blog/precision-agriculture-drones>
- [38] M. Costanzo, “What are the main technologies and applications of precision agriculture?,” Wikifarmer. Accessed: Jun. 06, 2024. [Online]. Available: <https://wikifarmer.com/what-are-the-main-technologies-and-applications-of-precision-agriculture/>
- [39] A. L. Samuel, “Some studies in machine learning using the game of checkers,” *IBM Journal of Research and Development*, vol. 44, no. 1.2, pp. 206–226, 2000.
- [40] T. M. Mitchell, “Machine learning,” *Engineering/Math*, vol. 1, p. 27, 1997.
- [41] E. Hossain, *Machine Learning Crash Course for Engineers*. Cham: Springer International Publishing, 2024. doi: 10.1007/978-3-031-46990-9.
- [42] Patanjali Kashyap, *Machine Learning for Decision Makers: Cognitive Computing Fundamentals for Better Decision Making*, 2nd ed. Berkeley, CA: Apress, 2024. doi: 10.1007/978-1-4842-9801-5.
- [43] “Self-supervised learning: The dark matter of intelligence.” Accessed: Apr. 25, 2024. [Online]. Available: <https://ai.meta.com/blog/self-supervised-learning-the-dark-matter-of-intelligence/>
- [44] F. Ros and R. Riad, *Feature and Dimensionality Reduction for Clustering with Deep Learning*. in *Unsupervised and Semi-Supervised Learning*. Cham: Springer Nature Switzerland, 2024. doi: 10.1007/978-3-031-48743-9.
- [45] B. Quinto, *Next-Generation Machine Learning with Spark: Covers XGBoost, LightGBM, Spark NLP, Distributed Deep Learning with Keras, and More*. Berkeley, CA: Apress, 2020. doi: 10.1007/978-1-4842-5669-5.
- [46] Y. Mansar, “Build Powerful Lightweight Models Using Knowledge Distillation,” *Medium*. Accessed: May 22, 2024. [Online]. Available: <https://towardsdatascience.com/build-powerful-lightweight-models-using-knowledge-distillation-618f69b569d9>
- [47] M. Negnevitsky, *Artificial intelligence: a guide to intelligent systems*, 3. ed. Harlow Munich: Addison-Wesley, 2011.
- [48] A. Kapoor, A. Gullì, and S. Pal, *Deep learning with TensorFlow and Keras: build and deploy supervised, unsupervised, deep, and reinforcement learning models*, Third edition. Birmingham Mumbai: Packt Publishing, 2022.

- [49] Nasrin Talkhi *et al.*, “Prediction of serum anti-HSP27 antibody titers changes using a light gradient boosting machine (LightGBM) technique,” *Scientific Reports*, vol. 13, no. 1, Aug. 2023, doi: 10.1038/s41598-023-39724-z.
- [50] NVIDIA, “Deep Learning.” Accessed: Apr. 26, 2024. [Online]. Available: <https://developer.nvidia.com/deep-learning>
- [51] A. H. Reynolds, “Convolutional Neural Networks (CNNs),” Anh H. Reynolds. Accessed: May 01, 2024. [Online]. Available: <https://anhreynolds.com/>
- [52] Laith Alzubaidi *et al.*, “Review of deep learning: concepts, CNN architectures, challenges, applications, future directions,” *Journal of Big Data*, vol. 8, no. 1, pp. 1–74, 2021, doi: 10.1186/s40537-021-00444-8.
- [53] T. T. Teoh, *Convolutional Neural Networks for Medical Applications*. in SpringerBriefs in Computer Science. Singapore: Springer Nature Singapore, 2023. doi: 10.1007/978-981-19-8814-1.
- [54] M. Parab and N. Mehendale, “Red Blood Cell Classification Using Image Processing and CNN,” *SN Computer Science*, vol. 2, Apr. 2021, doi: 10.1007/s42979-021-00458-2.
- [55] “Réseaux de neurones à classes multiples: Softmax | Machine Learning,” Google for Developers. Accessed: May 08, 2024. [Online]. Available: <https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax?hl=fr>
- [56] M. A. I. Khan, “Introduction to Softmax Classifier in PyTorch,” MachineLearningMastery.com. Accessed: May 08, 2024. [Online]. Available: <https://machinelearningmastery.com/introduction-to-softmax-classifier-in-pytorch/>
- [57] A. Munir, J. Kong, and M. A. Qureshi, *Accelerators for Convolutional Neural Networks*. Canada: IEEE PRESS, 2024.
- [58] B. Pragati, “Convolutional Neural Networks: Architectures, Types & Examples,” v7labs. Accessed: May 01, 2024. [Online]. Available: <https://www.v7labs.com/blog/convolutional-neural-networks-guide>
- [59] G. Abich, L. Ost, and R. Reis, *Early Soft Error Reliability Assessment of Convolutional Neural Networks Executing on Resource-Constrained IoT Edge Devices*. in Synthesis Lectures on Engineering, Science, and Technology. Cham: Springer Nature Switzerland, 2023. doi: 10.1007/978-3-031-18599-1.
- [60] M. Naved, V. A. Devi, L. Gaur, and A. A. Elngar, *IoT-enabled Convolutional Neural Networks: Techniques and Applications*, 1st ed. New York: River Publishers, 2023. doi: 10.1201/9781003393030.

- [61] M. Elgendy, *Deep learning for vision systems*. Shelter Island, NY: Manning Publications Co, 2020.
- [62] M. Akay *et al.*, “Deep Learning Classification of Systemic Sclerosis Skin Using the MobileNetV2 Model,” *IEEE Open J. Eng. Med. Biol.*, vol. 2, pp. 104–110, 2021, doi: 10.1109/OJEMB.2021.3066097.
- [63] F. Thiele, A. J. Windebank, and A. M. Siddiqui, “Motivation for using data-driven algorithms in research: A review of machine learning solutions for image analysis of micrographs in neuroscience,” 2023, doi: 10.1093/jnen/nlad040.
- [64] M. Swamynathan, *Mastering Machine Learning with Python in Six Steps: A Practical Implementation Guide to Predictive Data Analytics Using Python*, 2nd ed. India: Apress, 2019. doi: 10.1007/978-1-4842-4947-5.
- [65] D. Martinez, “Is Transfer Learning the final step for enabling AI in Aviation?,” *Datascience.aero*. Accessed: May 23, 2024. [Online]. Available: <https://datascience.aero/transfer-learning-aviation/>
- [66] “What is Transfer Learning?,” *GeeksforGeeks*. Accessed: May 23, 2024. [Online]. Available: <https://www.geeksforgeeks.org/ml-introduction-to-transfer-learning/>
- [67] N. Koenigstein, *Transformers in Action*, 7th ed. Manning Publications, 2024.
- [68] L. Tunstall, L. von Werra, and T. Wolf, *Natural Language Processing with Transformers*, 1st ed. United States of America: O’Reilly Media, 2022.
- [69] Z. Ralte and I. Kar, *Learn Python Generative AI*, 1st ed. India: BPB, 2024.
- [70] A. Shrotriya, A. K. Sharma, N. Pradhan, and P. Shukla, “A light weight Deep Convolutional Neural network model for plant disease identification,” in *2023 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*, 2023, pp. 191–196. doi: 10.1109/ICCIKE58312.2023.10131878.
- [71] H. Guan, C. Fu, G. Zhang, K. Li, P. Wang, and Z. Zhu, “A lightweight model for efficient identification of plant diseases and pests based on deep learning,” *Front. Plant Sci.*, vol. 14, p. 1227011, Jul. 2023, doi: 10.3389/fpls.2023.1227011.
- [72] W. Feng, Q. Song, G. Sun, and X. Zhang, “Lightweight Isotropic Convolutional Neural Network for Plant Disease Identification,” *Agronomy*, vol. 13, no. 7, p. 1849, Jul. 2023, doi: 10.3390/agronomy13071849.
- [73] B. Wang, C. Zhang, Y. Li, C. Cao, D. Huang, and Y. Gong, “An ultra-lightweight efficient network for image-based plant disease and pest infection detection,” *Precision Agric*, vol. 24, no. 5, pp. 1836–1861, Oct. 2023, doi: 10.1007/s11119-023-10020-0.

- [74] T. Sanida, M. V. Sanida, A. Sideris, and M. Dasygenis, “A Lightweight CNN Model for Tomato Crop Diseases on Heterogeneous Embedded System,” in *2023 12th International Conference on Modern Circuits and Systems Technologies (MOCASST)*, 2023, pp. 1–4. doi: 10.1109/MOCASST57943.2023.10176582.
- [75] X. Fang, T. Zhen, and Z. Li, “Lightweight Multiscale CNN Model for Wheat Disease Detection,” *Applied Sciences*, vol. 13, no. 9, p. 5801, May 2023, doi: 10.3390/app13095801.
- [76] P. K. Sahil Verma and J. P. Singh, “A Unified Lightweight CNN-based Model for Disease Detection and Identification in Corn, Rice, and Wheat,” *IETE Journal of Research*, vol. 0, no. 0, pp. 1–12, 2023, doi: 10.1080/03772063.2023.2181229.
- [77] Q. Huang *et al.*, “Knowledge Distillation Facilitates the Lightweight and Efficient Plant Diseases Detection Model,” *Plant Phenomics*, vol. 5, p. 0062, Jan. 2023, doi: 10.34133/plantphenomics.0062.
- [78] Y. Liu *et al.*, “High-Precision Tomato Disease Detection Using NanoSegmenter Based on Transformer and Lightweighting,” *Plants*, vol. 12, no. 13, p. 2559, Jul. 2023, doi: 10.3390/plants12132559.
- [79] P. S. Thakur, P. Khanna, T. Sheorey, and A. Ojha, “Explainable vision transformer enabled convolutional neural network for plant disease identification: PlantXViT.” arXiv, Jul. 16, 2022. Accessed: May 31, 2024. [Online]. Available: <http://arxiv.org/abs/2207.07919>
- [80] S. A. Wagle, R. Harikrishnan, S. H. M. Ali, and M. Faseehuddin, “Classification of Plant Leaves Using New Compact Convolutional Neural Network Models,” *Plants*, vol. 11, no. 1, p. 24, Dec. 2021, doi: 10.3390/plants11010024.
- [81] Y. Liu, G. Gao, and Z. Zhang, “Plant disease detection based on lightweight CNN model,” in *2021 4th International Conference on Information and Computer Technologies (ICICT)*, 2021, pp. 64–68. doi: 10.1109/ICICT52872.2021.00018.
- [82] M. H. Saleem, S. Khanchi, J. Potgieter, and K. M. Arif, “Image-Based Plant Disease Identification by Deep Learning Meta-Architectures,” *Plants*, vol. 9, no. 11, p. 1451, Oct. 2020, doi: 10.3390/plants9111451.
- [83] A. Munir, J. Kong, and M. A. Qureshi, “Accelerators for Convolutional Neural Networks”.
- [84] Devansh, “How does Batch Size impact your model learning,” Geek Culture. Accessed: Jun. 07, 2024. [Online]. Available: <https://medium.com/geekculture/how-does-batch-size-impact-your-model-learning-2dd34d9fb1fa>

- [85] “Parameters — LightGBM 4.4.0.99 documentation.” Accessed: Jun. 19, 2024. [Online]. Available: <https://lightgbm.readthedocs.io/en/latest/Parameters.html>
- [86] M. Abadi *et al.*, “TensorFlow, Large-scale machine learning on heterogeneous systems.” Nov. 2015. doi: 10.5281/zenodo.4724125.
- [87] K. Team, “Keras documentation: About Keras 3.” Accessed: Jun. 05, 2024. [Online]. Available: <https://keras.io/about/>
- [88] “Welcome to LightGBM’s documentation! — LightGBM 4.0.0 documentation.” Accessed: Jun. 05, 2024. [Online]. Available: <https://lightgbm.readthedocs.io/en/stable/>
- [89] “Using 🤖 transformers at Hugging Face.” Accessed: Jun. 05, 2024. [Online]. Available: <https://huggingface.co/docs/hub/transformers>
- [90] “PyTorch Foundation,” PyTorch. Accessed: Jun. 05, 2024. [Online]. Available: <https://pytorch.org/foundation>
- [91] J. Ansel *et al.*, “PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation,” *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, Apr. 2024. doi: 10.1145/3620665.3640366.
- [92] “onnx/onnx.” Open Neural Network Exchange, Jun. 05, 2024. Accessed: Jun. 05, 2024. [Online]. Available: <https://github.com/onnx/onnx>
- [93] “ONNX Runtime | Inference.” Accessed: Jun. 05, 2024. [Online]. Available: <https://onnxruntime.ai/inference>
- [94] “Streamlit Docs.” Accessed: Jun. 05, 2024. [Online]. Available: <https://docs.streamlit.io/>
- [95] D. P. Hughes and M. Salathé, “An open access repository of images on plant health to enable the development of mobile disease diagnostics”.
- [96] K. Team, “Keras documentation: Image augmentation layers.” Accessed: Jun. 19, 2024. [Online]. Available: https://keras.io/api/layers/preprocessing_layers/image_augmentation/
- [97] “Google Colab.” Accessed: Jun. 18, 2024. [Online]. Available: <https://research.google.com/colaboratory/faq.html>
- [98] “Project Jupyter Documentation — Jupyter Documentation 4.1.1 alpha documentation.” Accessed: Jun. 18, 2024. [Online]. Available: <https://docs.jupyter.org/en/latest/>
- [99] “Anaconda vs Python Programming Explained With Differences,” BairesDev. Accessed: Jun. 18, 2024. [Online]. Available: <https://www.bairesdev.com/blog/anaconda-vs-python-programming/>

- [100] “Quick start guide | PyCharm,” PyCharm Help. Accessed: Jun. 18, 2024. [Online]. Available: <https://www.jetbrains.com/help/pycharm/quick-start-guide.html>
- [101] “What is Kaggle?” Accessed: Jun. 19, 2024. [Online]. Available: <https://www.datacamp.com/blog/what-is-kaggle>
- [102] “torchvision — Torchvision 0.18 documentation.” Accessed: Jun. 18, 2024. [Online]. Available: <https://pytorch.org/vision/stable/index.html>
- [103] Ashish, “15 Most Important Features of Scikit-Learn!,” Analytics Vidhya. Accessed: Jun. 18, 2024. [Online]. Available: <https://www.analyticsvidhya.com/blog/2021/07/15-most-important-features-of-scikit-learn/>
- [104] “Pandas | Python Library - Mode,” Mode Resources. Accessed: Jun. 19, 2024. [Online]. Available: <https://mode.com/python-tutorial/libraries/pandas/>
- [105] “What is NumPy? — NumPy v2.1.dev0 Manual.” Accessed: Jun. 19, 2024. [Online]. Available: <https://numpy.org/devdocs/user/whatisnumpy.html>
- [106] M. L. Waskom, “seaborn: statistical data visualization,” *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, Apr. 2021, doi: 10.21105/joss.03021.
- [107] “OpenCV Tutorial in Python,” GeeksforGeeks. Accessed: Jun. 19, 2024. [Online]. Available: <https://www.geeksforgeeks.org/opencv-python-tutorial/>
- [108] Melanie, “Pillow: How to process images with Python,” Data Science Courses | DataScientest. Accessed: Jun. 19, 2024. [Online]. Available: <https://datascientest.com/en/pillow-how-to-process-images-with-python>
- [109] “Introduction to Matplotlib,” GeeksforGeeks. Accessed: Jun. 19, 2024. [Online]. Available: <https://www.geeksforgeeks.org/python-introduction-matplotlib/>
- [110] “Tomato Yellow Leaf Curl Virus | NC State Extension Publications.” Accessed: Jun. 18, 2024. [Online]. Available: <https://content.ces.ncsu.edu/tomato-yellow-leaf-curl-virus>
- [111] “Tomato Leaf Mold | Cornell Vegetables.” Accessed: Jun. 18, 2024. [Online]. Available: <https://www.vegetables.cornell.edu/pest-management/disease-factsheets/tomato-leaf-mold/>