

الجمهورية الجزائرية الديمقراطية الشعبية

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

وزارة التعليم العالي والبحث العلمي

Ministry of Higher Education and Scientific Research



Saad Dahlab Blida 1 University
Institute of Aeronautics
and Space Studies
Department of Space Studies



In order to obtain the Master's degree in Aeronautics

Option: Space Telecommunications

THEME

Early Detection of Citrus Diseases Using Deep Learning:
An AI Framework Intended for UAV-Based Smart
Farming

Under the supervision of:

❖ **Dr. TAHRAOUI Sofiane**
Senior Lecturer -A-

A thesis submitted by:

❖ **Mr. FEGAS Abdessemed**
❖ **Mr. BENBOUZID Ahmed Nader**

Examined by :

❖ **Dr. KRIM Mohammed Senior Lecturer -B- (President of the jury)**
❖ **Dr. AZMEDROUB Boussad Senior Lecturer -B- (Examinator of the jury)**

Promotion: 2024 / 2025

Thanks

This thesis would not have been possible without the help of many. Above all, it is an opportunity to express our gratitude to Allah, the Almighty, for granting us the courage and willpower to accomplish this work.

We extend our sincere thanks and special recognition to our final-year project supervisor, Dr. TAHRAOUI Sofiane, for his valuable guidance throughout this thesis, as well as for his efforts and attentive support.

We would also like to thank our parents for their unwavering support and constant encouragement throughout our academic journey; this success is undoubtedly the result of their many years of sacrifice and dedication to our education.

We warmly thank the jury members for agreeing to review our work and provide constructive feedback.

Our gratitude goes as well to the teaching and administrative staff of the Aeronautics Department.

Finally, we express our heartfelt thanks and deep appreciation to all the teaching and administrative staff of the Aeronautics Department, our friends, and everyone who contributed, directly or indirectly, to the completion of this thesis.

Abstract

Citrus diseases pose a significant threat to global citrus production, affecting yield, fruit quality, and economic sustainability. Traditional detection methods are often time-consuming, labor-intensive, and prone to human error. Recent advances in Artificial Intelligence (AI), particularly in machine learning and computer vision, have enabled the development of automated and accurate systems for early disease detection. AI-powered models can analyze images of citrus leaves, fruits, and trees to identify diseases such as Citrus Greening (HLB), canker, and black spot with high precision. Techniques like Convolutional Neural Networks (CNNs) are commonly used to classify disease symptoms from visual data, while integrated systems leverage remote sensing and Internet of Things (IoT) devices for real-time monitoring. The purpose of this study is to develop and evaluate an AI-based detection framework in a drone for identifying major citrus diseases at an early stage, aiming to improve diagnostic accuracy, reduce crop losses, and support timely disease management in citrus orchards. This approach seeks to enhance agricultural productivity and sustainability through the adoption of intelligent, data-driven technologies.

ملخص

تشكل أمراض الحمضيات تهديدًا كبيرًا لإنتاج الحمضيات على مستوى العالم، حيث تؤثر على الغلة وجودة الثمار والاستدامة الاقتصادية. غالبًا ما تكون طرق الكشف التقليدية بطيئة وتتطلب جهدًا بشريًا كبيرًا، كما أنها عرضة للأخطاء. وقد مكنت التطورات الحديثة في تقنيات الذكاء الاصطناعي، وخاصة في تعلم الآلة والرؤية الحاسوبية، من تطوير أنظمة مؤتمنة ودقيقة للكشف المبكر عن الأمراض. تستطيع النماذج المعتمدة على الذكاء الاصطناعي تحليل صور لأوراق الحمضيات وثمارها وأشجارها لتحديد أمراض مثل مرض البقع الأسود، ومرض التقرح، ومرض التدهور السريع (التخضير/هوانغلونينغ) بدقة عالية. وتستخدم الشبكات العصبية الالتفافية (CNNs) بشكل شائع لتصنيف أعراض الأمراض من البيانات البصرية، كما تعتمد الأنظمة المتكاملة على الاستشعار عن بُعد وأجهزة إنترنت الأشياء (IoT) للمراقبة في الوقت الفعلي. وتهدف هذه الدراسة إلى تطوير وتقييم إطار عمل للكشف عن أمراض الحمضيات باستخدام الذكاء الاصطناعي في الطائرات بدون طيار، بهدف تحسين دقة التشخيص، وتقليل الخسائر في المحاصيل، ودعم الإدارة الفعالة للأمراض في بساتين الحمضيات. وتسعى هذه المقاربة إلى تعزيز الإنتاجية الزراعية والاستدامة من خلال تبني تقنيات ذكية تعتمد على البيانات.

Résumé

Les maladies des agrumes représentent une menace majeure pour la production mondiale, affectant le rendement, la qualité des fruits et la durabilité économique. Les méthodes traditionnelles de détection sont souvent longues, coûteuses en main-d'œuvre et sujettes à l'erreur humaine. Les avancées récentes en intelligence artificielle (IA), notamment dans l'apprentissage automatique et la vision par ordinateur, ont permis le développement de systèmes automatisés et précis pour la détection précoce des maladies. Les modèles basés sur l'IA peuvent analyser des images de feuilles, de fruits et d'arbres d'agrumes afin d'identifier avec précision des maladies telles que le verdissement des agrumes (HLB), le chancre et la tache noire. Des techniques comme les réseaux de neurones convolutifs (CNN) sont couramment utilisées pour classer les symptômes des maladies à partir de données visuelles, tandis que des systèmes intégrés exploitent la télédétection et les dispositifs de l'Internet des objets (IoT) pour une surveillance en temps réel. L'objectif de cette étude est de développer et d'évaluer un cadre de détection basé sur l'intelligence artificielle dans une conception de drone pour identifier précocement les principales maladies des agrumes, dans le but d'améliorer la précision diagnostique, de réduire les pertes de récolte et de favoriser une gestion rapide et efficace des maladies. Cette approche vise à renforcer la productivité agricole et la durabilité grâce à l'adoption de technologies intelligentes basées sur les données .

Table of Contents

Abstract	
List of Figures	
List of Tables	
List of Acronyms	
General Introduction.....	1

Chapter I : Generalities

I.1. Introduction.....	5
I.2. Citrus Black Spot (CBS).....	5
I.3. Citrus Greening (HLB)	6
I.4. Citrus Canker	7
I.5. Comparison of Citrus Greening , Black Spot , and Canker	7
I.5.1. Causal Agents and Pathogen Type.....	8
I.5.2. Vectors and Modes of Transmission.....	8
I.5.3. Symptoms and Disease Progression.....	9
I.5.4. Management and Control Strategies	9
I.5.5. Host Range and Affected Tissues	10
I.5.6. Geographic Distribution and Importance.....	10
I.6. Conclusion.....	11

Chapter II : Artificial Intelligence and Image Processing

II.1. Introduction.....	13
II.2. Artificial Intelligence	13
II.2.1. Overview of Artificial Intelligence	13
II.2.1.1. Definition of Artificial Intelligence	13
II.2.1.2. The Objective and General Impact of Artificial Intelligence	14
II.2.2. History of Artificial Intelligence	15
II.3. Application Areas of Artificial Intelligence	16
II.3.1. Fields of Application	16
II.3.2. Research Areas	17
II.4. UAV-Based Smart Farming	18
II.4.1. Implementation of UAVs in Agriculture	18
II.4.1.1. Crop Monitoring and Health Assessment	18
II.4.1.2. Soil and Field Analysis	19
II.4.1.3. Irrigation Management	19
II.4.1.4. Crop Counting and Yield Prediction	19
II.4.1.5. Livestock Monitoring	19
II.4.2. Problems and Challenges of UAVs in Agriculture	19
II.4.2.1. High Initial Cost	19
II.4.2.2. Technical Complexity	20
II.4.2.3. Limited Flight Time and Range	20
II.4.2.4. Data Processing Challenges.....	20
II.4.2.5. Regulatory Restrictions.....	20
II.4.2.6. Privacy and Security Concerns... ..	20

II.4.2.7. Environmental and safety Risks	20
II.5. CNN.....	20
II.6. Image Processing... ..	21
II.6.1. Definition of Image Processing... ..	21
II.7. Transfer Learning... ..	22
II.8. Machine Learning... ..	24
II.9. VGG-16... ..	25
II.9.1. VGG Architecture.....	26
II.9.2. The Differences Between VGG-16 and VGG-19 Networks... ..	27
II.10. Keras Function... ..	28
II.11. TensorFlow Function... ..	28
II.12. Image Data Generator... ..	29
II.13. Flatten and Dense.....	30
II.14. Earlystopping... ..	31
II.15. Matplotlib.pyplot.....	32
II.16. MobileNet... ..	32
II.17. DenseNet 201.....	34
II.18. Load model... ..	36
II.19. Sklearn... ..	36
II.20. Confusion Matrix.....	37
II.21. Conclusion... ..	38

Chapter III : Simulation and Results

III.1. Introduction.	40
III.2. Platform used for implementation	40
III.2.1. Software.....	40
III.2.1.1. Python 3.11.3.....	40
III.2.1.2. Jupyter 6.5.4... ..	40
III.3. Working Principal.	41
III.3.1. Data Input Gathering the Examples	41
III.3.1.1. Principle.....	41
III.3.1.2. Purpose	41
III.3.2. Processing (Getting Ready to Learn)	41
III.3.2.1. Principle.....	41
III.3.2.2. Purpose	41
III.3.3. Model Training (The Learning Phase)	42
III.3.3.1. Principle.....	42
III.3.3.2. Process.....	42
III.3.3.3. Outcome	42
III.3.4. Output Prediction (Identifiying new Cases)	42
III.3.4.1. Principle.....	42
III.3.4.2. Process... ..	42
III.3.4.3. Result.....	43
III.4. The Loss and Accuracy of the Validation and the Training During the Training.....	44
III.4.1. Validation	44
III.4.2. Train... ..	44
III.4.2.1. Loss During Training	45

III.4.2.1.1. Train Loss (Blue)	45
III.4.2.1.2. Validation Loss (Orange)	45
III.4.2.2. Accuracy During Training.....	45
III.4.2.2.1. Train Accuracy (Blue).....	45
III.4.2.2.2. Validation Accuracy Orange	46
III.5. Test... ..	47
III.5.1. Test Loss... ..	47
III.5.2. Test Accuracy... ..	47
III.5.3. Interpretation	48
III.5.3.1. Precision	48
III.5.3.2. Recall.....	48
III.5.3.3. F-1 Score	49
III.5.3.4. Macro Avg and Weighted Avg	49
III.5.4. Confusion Matrix	50
III.5.4.1. Interpretation	50
III.5.5. Model Prediction Output... ..	50
III.6. Conclusion.....	51
General Conclusion	52
References	53

Figures list

Figure 1: Citrus Black Spot (CBS)	6
Figure 2: Citrus Greening (HLB)	7
Figure 3: Citrus Canker	8
Figure 4: Photos on the use of virtual reality and artificial intelligence in research [7]	18
Figure 5: simple Architecture of CNN [5]	21
Figure 6: Block Diagram of transfer-learning [6]	23
Figure 7: Traditional Learning vs Transfer Learning [9]	24
Figure 8: VGG-16 architecture [12]	25
Figure 9: VGG-16 architecture Map [12]	26
Figure 10: Summary Diagram of the Main Functions Through Keras	28
Figure 11: DenseNet 201 Architecture	35
Figure 12: Functioning Diagrammed of the Machine (Model)	43

Tables

Table 1: Comparison between VGG19 and VGG16 [12]	27
Table 2: Classification Report	47

List of Acronyms

- ❖ **(ILSVRC)** : ImageNet Large Scale Visual Recognition Challenge
- ❖ **1D** : 1 Dimension
- ❖ **2D** : 2 Dimensions
- ❖ **AI** : Artificial Intelligence
- ❖ **API** : Application Programming Interface
- ❖ **CBS** : Citrus Black Spot
- ❖ **CNN** : Convolutional Neural Network
- ❖ **CNTK** : Computational Network Toolkit
- ❖ **CPU** : Central Processing Unit
- ❖ **DL** : Deep Learning
- ❖ **EfficientNet** : Efficient Neural Network
- ❖ **ELISA** : Enzyme-Linked Immunosorbent Assay
- ❖ **FCL** : Fully Connected Layer
- ❖ **FLOPs** : Floating Point Operations Per second
- ❖ **GIS** : Geographic Information Systems
- ❖ **GPU** : Graphics Processing Unit

- ❖ **Grad-CAM** : Gradient-Weighted Class Activation Mapping
- ❖ **GUI** : Graphical User Interface
- ❖ **HDF** : Hierarchical Data Format
- ❖ **HLB** : Huanglongbing
- ❖ **HSV** : Hue Saturation Value
- ❖ **Iot** : Internet Of Things
- ❖ **IPC** : Inter-Process Communication
- ❖ **LSTM** : Long Short-Term Memory
- ❖ **MINIX** : Mini UNIX
- ❖ **ML** : Machine Learning
- ❖ **NAS** : Neural Architecture Search
- ❖ **PCR** : Polymerase Chain Reaction
- ❖ **PNG** : Portable Network Graphic
- ❖ **QNX** : Quick UNIX
- ❖ **R-CNN** : Region-based Convolutional Neural Network
- ❖ **ReLU** : Rectified Linear Unit
- ❖ **ResNet** : Residual Network
- ❖ **RGB** : Red Green Blue
- ❖ **RMSprop** : Root Mean Square Propagation
- ❖ **SE** : Squeeze-and-Excitation
- ❖ **SGD** : Stochastic Gradient Descent
- ❖ **SVC** : Support Vector Classification
- ❖ **TKAgg** : Tkinter + Anti-Grain Geometry
- ❖ **TPU** : Tensor Processing Unit
- ❖ **UAVs** : Unmanned Aerial Vehicles
- ❖ **US** : United State
- ❖ **VGGNet** : Visual Geometry Group Network
- ❖ **ViTs** : Vision Transformers
- ❖ **YOLO** : You Only Look Once

General introduction

Agriculture plays a fundamental role in driving the economic growth of nations across the globe. Among various agricultural products, citrus fruits hold particular importance due to their health benefits, including immune-boosting properties and their ability to combat infections. However, citrus crops are vulnerable to a range of diseases that can significantly reduce yield and cause substantial financial losses for farmers.

Early and accurate detection of citrus diseases is critical for effective disease management, reducing economic impact, and promoting sustainable agricultural practices. Traditional disease detection methods typically rely on laboratory testing and expert analysis, which can be time-consuming, labor-intensive, and prone to human error. Delays in diagnosis and intervention can lead to the rapid spread of diseases, further exacerbating losses.

There is a pressing need for automated systems capable of early disease detection, classification, and the recommendation of preventive measures. Recent advancements in Artificial Intelligence (AI), particularly in Machine Learning (ML), offer promising solutions for the automatic and reliable detection of plant diseases. By integrating image processing with ML techniques [1], these systems can identify diseases with high precision using images of infected citrus plants.

Machine learning models, when properly trained, can predict diseases based on unknown input images by analyzing visual patterns and extracting key features. These features capture important statistical characteristics from the images, allowing the models to determine the type and severity of the disease.

The effectiveness of ML-based disease detection largely depends on:

- The availability of a comprehensive and high-quality dataset,
- The resolution and clarity of the input images,
- The optimization of relevant features used for training.

Feature optimization plays a vital role in enhancing model performance by selecting the most informative variables from image data. Using robust feature selection and extraction techniques helps maintain high classification accuracy while improving computational efficiency. This not only reduces processing time but also contributes to the development of more reliable and interpretable detection systems.

This study proposes a machine learning framework that integrates feature optimization techniques for the accurate identification of citrus diseases.

By isolating the most discriminative features from citrus leaf images, the approach aims to improve both the accuracy and efficiency of disease classification models. This work evaluates multiple machine learning classifiers, feature selection methods, and performance metrics to develop a consistent and reliable detection system.

In recent years, ML-based approaches have emerged as powerful tools for automating citrus disease diagnosis. These techniques combine advanced image processing, feature extraction, and classification to detect symptoms in leaves, fruits, and stems. Incorporating feature optimization further enhances model accuracy, reduces computational complexity, and delivers a scalable, cost-effective solution for real-world agricultural use.

The study and analysis can be done through two methods, the traditional method and the technology method.

Traditional methods for citrus disease detection largely depend on visual inspection carried out by farmers or agricultural experts. An overview of these detection techniques is illustrated in Figure 1. Despite being widely used, conventional diagnostic approaches face several critical limitations:

- **Dependence on Expert Knowledge:** The accuracy of manual disease identification heavily depends on the expertise and experience of the evaluator. Inaccurate diagnoses by untrained or less experienced personnel can lead to inappropriate treatments and ineffective disease management.
- **Labor-Intensive and Time-Consuming:** Conducting field surveys and visually inspecting each plant requires considerable time and physical effort, making it difficult to perform frequent or large-scale monitoring.
- **Delayed Detection:** Manual methods are often unable to identify diseases at their early stages. By the time symptoms become visually apparent, the disease may have already progressed and spread, reducing the effectiveness of any remedial actions.

- **Limitations of Laboratory Testing:** While laboratory-based techniques such as Polymerase Chain Reaction (PCR) [2] and Enzyme-Linked Immunosorbent Assay (ELISA) [2] offer high accuracy, they are resource-intensive. These methods require specialized equipment, trained personnel, and considerable time and cost, them impractical for or large-scale deployment in the field.

In contrast to traditional techniques, machine learning (ML)-based approaches offer several significant advantages for citrus disease detection:

- **Automated and Scalable Analysis:** ML models are capable of analyzing large volumes of data efficiently, enabling rapid and automated disease detection without the need for continuous human oversight.
- **Enhanced Accuracy and Reliability:** By eliminating subjectivity and human error, especially through deep learning and optimized feature selection, ML models deliver more accurate and consistent classification results.
- **Early-Stage Detection:** Machine learning systems can identify diseases at an early stage by detecting subtle patterns or visual indicators that may not be noticeable to the human eye, allowing for timely and effective intervention.
- **Cost-Effective and User-Friendly Deployment:** Once trained, ML models can be implemented on low-cost mobile or edge devices, making the technology accessible and practical for farmers, even those with limited technical expertise.

This work is divided into three chapters

The first chapter: presents generalities about the citrus diseases that we choosed.

The second chapter: presents an explanation about the Artificial Intelligence and image processing.

The third chapter: presents the analysis of results.

Chapter I :

Generalities

I.1. Introduction

Citrus fruits including oranges, lemons, mandarins, limes, and grapefruits are cultivated in over 140 countries and represent a vital component of global agriculture and economy. Valued for their nutritional benefits, culinary versatility, and high market demand, citrus fruits contribute significantly to food security and rural development, especially in tropical and subtropical regions. However, this vibrant sector faces serious challenges due to the widespread incidence of citrus diseases.

Citrus plants are vulnerable to a wide variety of pathogens, including fungi, bacteria, viruses, viroids, and phytoplasmas [2] as well as damage from nematodes and abiotic stresses (such as nutrient deficiencies, drought, or frost) [2]. These diseases can affect every part of the plant from roots to leaves, flowers, and fruits and often result in yield reduction, fruit deformities, tree decline, and even death.

The economic impact of citrus diseases is profound. Infected orchards often suffer from reduced productivity, increased cost of chemical control, and strict quarantine regulations that limit international trade. Some pathogens can spread rapidly across entire regions, leading to large-scale destruction and loss of biodiversity in commercial and traditional citrus varieties.

I.2. Citrus Black Spot (CBS)

Citrus Black Spot (CBS) is a fungal disease caused by *Phyllosticta citricarpa* (previously known as *Guignardia citricarpa*) that affects citrus plants [3], leading to a decline in both fruit quality and yield. The pathogen produces noticeable lesions on the fruit rind, significantly reducing the fruit's visual appeal and making it unsuitable for fresh markets. Among the most susceptible citrus types are Navel and Valencia oranges, lemons, and grapefruits, while sour orange and its variants show moderate resistance. Notably, rough lemons and acid limes do not typically exhibit symptoms of CBS.

The disease presents itself in several forms, including hard spots, cracked spots, false melanose, and virulent spots, each with its own distinct appearance.

The most characteristic and widely recognized symptom is the hard spot small, sunken, round lesions with grayish centers and reddish-brown borders. CBS thrives under specific climatic conditions, particularly in cool and moist environments during the winter season. Because of its potential to spread and affect trade, CBS is classified as a quarantine disease in many regions. Effective management includes the application of fungicides, especially copper-based or strobilurin compounds, along with careful monitoring and orchard sanitation practices.



Figure 1: Citrus Black Spot (CBS)

I.3. Citrus Greening Disease (Huanglongbing - HLB)

Citrus Greening Disease, also known as Huanglongbing (HLB) [4], is one of the most devastating diseases affecting citrus crops worldwide. It is caused by the bacterium *Candidatus Liberibacter* (primarily asiaticus) [4], which is transmitted by the Asian citrus psyllid (*Diaphorina citri*) [3]. The disease disrupts the flow of nutrients within the plant, leading to symptoms such as yellowing of leaves, shoot dieback, misshapen and bitter-tasting fruits, and a general decline in tree health. Affected fruits often remain partially green, which gives the disease its name. Over time, infected trees produce less fruit and may eventually die. There is currently no cure for citrus greening, making it a major threat to citrus industries in Asia, the Americas, and parts of Africa.

Management strategies focus on controlling the psyllid vector, removing infected trees, and using disease-free planting material. HLB's rapid spread and severe impact have made it a key target for global citrus research and quarantine efforts.



Figure 2: Citrus Greening (HLB)

I.4. Citrus Canker

Citrus canker is a highly contagious bacterial disease caused by *Xanthomonas citri* subsp [3]. *citri*, affecting a wide range of citrus species, especially grapefruit, Mexican lime, and sweet orange. It produces characteristic raised, corky lesions with yellow halos on leaves, stems, and fruit, leading to defoliation, fruit blemishes, and premature fruit drop. The pathogen enters through stomata or wounds and spreads rapidly via wind-driven rain, contaminated tools, and infected plant material, especially under warm, humid conditions. The disease significantly impacts citrus production and trade due to reduced fruit quality and yield, as well as strict quarantine measures. Management involves integrated approaches including the use of certified disease-free nursery stock, copper-based bactericides, removal of infected trees, sanitation practices, and, where available, planting of more resistant cultivars. Despite control efforts, citrus canker remains a major threat to the global citrus industry due to its persistence and the lack of fully resistant commercial varieties.



Figure 3: Citrus Canker

I.5. Comparison of Citrus Greening, Black Spot, and Canker

The three diseases Citrus Greening (HLB), Citrus Black Spot (CBS), and Citrus Canker differ significantly in terms of causal agents, modes of transmission, plant symptoms, and management strategies, each requiring a tailored approach to detection and control.

I.5.1. Causal Agents and Pathogen Type

Citrus greening is caused by a bacterium from the genus *Candidatus Liberibacter*, most notably *Candidatus Liberibacter asiaticus* [3]. This pathogen is phloem-limited, meaning it resides inside the vascular tissues of the plant, blocking nutrient transport and resulting in systemic decline. In contrast, citrus black spot is due to a fungus, *Phyllosticta citricarpa* [3], which primarily infects external tissues like the fruit rind and leaf surfaces, and reproduces via fungal spores. Meanwhile, citrus canker is caused by another bacterium, *Xanthomonas citri* subsp [4]. *Citri*, which colonizes intercellular spaces of leaf, fruit, and twig tissues, forming characteristic lesions. The fundamental difference between these pathogens fungal vs. bacterial, vascular vs. surface-localized determines how they affect the tree and how they spread.

I.5.2. Vectors and Modes of Transmission

A major distinction lies in how these pathogens are spread. Citrus greening relies on insect vectors, particularly the Asian citrus psyllid (*Diaphorina citri*) and African citrus psyllid (*Trioza erytreae*) [4]. These tiny insects acquire the bacteria while feeding on infected trees and inject it into healthy trees during feeding, making vector control a top priority. By

contrast, citrus black spot does not have an insect vector. Its fungal spores are dispersed mainly by rain splash, wind, and through infected leaf litter, which makes humidity and rainfall important factors in its epidemiology.

Citrus canker spreads through wind-driven rain, but also mechanical contact such as tools, equipment, workers' clothing, and contaminated nursery stock. These differing transmission modes highlight why greening demands psyllid monitoring and eradication campaigns, while black spot and canker require environmental management and strict hygiene protocols.

I.5.3. Symptoms and Disease Progression

The symptoms of each disease also vary widely in both appearance and severity. In citrus greening, leaves show blotchy mottling that is often asymmetrical, a key diagnostic feature. Fruits become small, green, misshapen, and bitter, often with aborted seeds and delayed color change. The infection progresses internally, eventually killing the tree. In black spot, the damage is mostly cosmetic in nature, affecting the outer fruit rind with distinct necrotic lesions ranging from "hard spot" (sunken black lesions) to speckled or freckled blotches leading to early fruit drop and economic loss due to downgraded fruit. The disease is not systemic and doesn't kill trees. Citrus canker, on the other hand, causes raised, corky lesions surrounded by a yellow halo on leaves, fruit, and stems. These lesions may merge into large blisters, leading to defoliation, twig dieback, and reduced fruit quality, but the disease does not move systemically in the plant like HLB does.

I.5.4. Management and Control Strategies

Control measures reflect these biological differences. Citrus greening is the most difficult to manage because it is systemic and incurable once a tree is infected, it must often be removed. Management relies on early detection, mass removal of infected trees, biological and chemical control of psyllid populations, and the use of certified disease-free nursery stock. Research into resistant or tolerant rootstocks and scions is ongoing but no commercial solution has yet proven fully effective.

For citrus black spot, control is more feasible through protective fungicide sprays especially strobilurins and copper-based fungicides applied at key periods during the fruit development stage. Cultural practices like pruning for better airflow, sanitation of leaf litter, and use of resistant cultivars can reduce spore load and delay onset.

Citrus canker control is similarly complex but involves copper-based bactericides, strict quarantine, and sanitation measures. In many regions, infected trees are destroyed to prevent spread. The use of windbreaks helps reduce rain splash transmission. Unlike HLB, canker can survive on the plant surface, making tool disinfection and worker hygiene especially important.

I.5.5. Host Range and Affected Tissues

All three diseases affect various parts of the tree but with different scopes. Citrus greening affects the entire tree, disrupting vascular flow and eventually leading to tree death. In black spot, the disease is primarily superficial, affecting fruit rinds and leaves, but not the fruit's internal quality. Citrus canker affects leaves, fruit, and young twigs, causing lesions that compromise the market appearance and potentially reduce yield, but it doesn't directly kill the tree unless compounded by other stresses.

I.5.6. Geographic Distribution and Economic Importance

From a geographic standpoint, citrus greening is now present in all major citrus-growing regions including Asia, North and South America, and parts of Africa. It is the most feared disease in the citrus industry due to its devastating long-term effects and lack of cure. Citrus black spot is more localized widespread in South Africa, Brazil, and parts of the United States (notably Florida) but is heavily regulated by international trade barriers, making it a major concern for exporters. Citrus canker, originally from Asia, has spread to South America and the southern U.S., causing severe outbreaks in areas with warm, humid climates. Countries that are canker-free enforce strict quarantine protocols to prevent introduction.

I.6. Conclusion

Citrus diseases such as greening (HLB), black spot, and canker represent serious threats to global citrus production, each with distinct symptoms, transmission methods, and management challenges. Traditional detection approaches, although widely used, suffer from key limitations including labor intensity, dependence on expert judgment, and delayed disease identification. These issues are particularly pronounced in large-scale agricultural settings, where visual inspections and manual diagnostics become impractical.

To overcome these challenges, the introduction of modern technologies—especially Artificial Intelligence (AI) and Remote Sensing (RS) through drone platforms—has become essential. These innovations enable scalable, precise, and real-time disease detection, significantly enhancing the efficiency of citrus orchard monitoring. By combining advanced image processing with machine learning, AI systems can detect diseases in early stages, reducing crop loss and guiding effective intervention strategies.

This chapter has laid the foundational understanding of the biological, economic, and geographical dimensions of citrus diseases. In the next chapter, we will explore in detail how AI and image processing technologies can be applied to detect these diseases with high accuracy, offering a transformative approach to sustainable and intelligent citrus farming.

Chapter II : Artificial Intelligence and Image Processing

II.1. Introduction

Artificial Intelligence (AI) and image processing are closely related fields that have seen remarkable advancements in recent years. AI, particularly through subfields such as deep learning and machine learning, has revolutionized the way images are processed and analyzed.

II.2. Artificial Intelligence

II.2.1. Overview of Artificial Intelligence

II.2.1.1. Definition of Artificial Intelligence

The word "intelligence" is derived from the Latin "intellegere" or "intelligere." It is worth noting that the Internet, by amplifying the mechanisms of information reinforcement among humans, has greatly contributed to establishing the concept of collective intelligence. Intelligence is the dynamic ability to make inferences from stimuli, derive abstractions, and create a language that enables the naming, exchange, and connection of these abstractions. Intelligence has made it possible to define the concept of context, thereby clarifying that relationships are not necessarily repetitive. These capabilities are what distinguish humans from other mammals. Not only can a dog not say "tomorrow," but it is also likely that the concept of "tomorrow" is not developed within its cognitive abilities. An example of artificial intelligence was proposed by British mathematician Alan Turing in 1950 .

II.2.1.2. The Objective and General Impact of Artificial Intelligence

The goal of Artificial Intelligence (AI) is to design systems capable of replicating human behavior in reasoning activities. AI aims to model intelligence as a phenomenon, much like physics, chemistry, or biology seek to model other natural phenomena.

Artificial Intelligence has a profound and multifaceted impact on various aspects of society, the economy, and daily life, particularly in areas such as the economy and employment, healthcare and medicine, transportation and logistics, education, security and defense, environment, and agriculture, among others. In summary, AI holds immense potential to positively transform many aspects of society, but it is essential to address the associated ethical and social challenges to ensure the responsible and equitable deployment of these technologies.

II.2.2. History of Artificial Intelligence

- **Gestation of AI (1943–1955):** During this period, the first foundational works that can be considered the beginning of artificial intelligence were carried out, even though the term "AI" did not yet exist. Notably, McCulloch and Pitts introduced a model of artificial neurons in 1943.

A few years later, Hebb proposed a rule for modifying connections between neurons, and Minsky and Edmonds built the first neural network. It was also during this time that Turing published his famous paper introducing the Turing Test [8].

- **Birth of AI (1956):** In this year, a small group of computer scientists interested in the study of intelligence gathered for a conference that marked the official birth of AI as a field.
- **Rising Hopes (1952–1969):** This was a highly active period for the young field of AI. A large number of programs were developed to solve a wide variety of problems. The Logic Theorist (by Newell and Simon) and the Geometry Theorem Prover (by Gelernter) were able to prove certain mathematical theorems—already known, but sometimes with more elegant proofs [8].
- **Early Disappointments (1966–1973):** During these years, it became increasingly clear that the predictions made by AI researchers had been overly optimistic. A notable example of this was the failure of machine translation to meet expectations.

- **Expert Systems (1969–1979):** The first expert system, called DENDRAL, was created in 1969 for the specialized task of determining a molecule's structure based on its chemical formula and mass spectrometry results.
- **AI in Industry (1980–present):** In the early 1980s, the company DEC began using an expert system to assist in configuring computer systems, saving tens of millions of dollars annually.
- **Modern AI (1987–present):** Over time, artificial intelligence has become an increasingly rigorous and formal scientific discipline. Most modern approaches are based on mathematical theories or experimental studies rather than intuition, and they are more frequently applied to real-world problems [8].

II.3. Application Areas of Artificial Intelligence

II.3.1. Fields of Application

- **Expert systems** (used in medicine, financial analysis, and device configuration), are employed for tasks such as diagnostics, monitoring, or troubleshooting in industrial settings [7].
- **Robotics and CAD/CAM (Computer-Aided Manufacturing):** Involves the introduction of robots that gather information through sensors or cameras, allowing them to navigate and operate in diverse environments.
- **Language understanding and machine translation:** Development of natural language interfaces, enabling users to query databases in English or French [7].
- **Pattern recognition** (speech, image, and handwriting recognition): For example, IBM uses a voice interface that recognizes 20,000 English business-related words and displays them on screen [7].

- **Learning:** Refers to creating programs capable of generating their own knowledge by modifying themselves based on their successes and failures.
- **Artificial emotion:** Refers to research on emotional computing, such as the work of Rosalind Picard on emotion recognition and modeling.

II.3.2. Research Areas

- **Machine Learning:** This process includes analytical learning systems, which aim to analyze and reformat existing knowledge into a more efficient or "operational" form. Synthetic learning systems, on the other hand, aim to discover fundamentally new knowledge [7].
- **Virtual Reality:** This field offers new forms of interaction between humans and machines. The advent of more powerful computers with advanced graphic capabilities, combined with visualization and interaction devices (such as headsets, gloves, etc.), makes it possible to provide the necessary sensory information (Figure 4).
- **Pattern Recognition:** Research in this field focuses on automating the identification of typical situations in terms of perception. Its methods have numerous applications, including computer vision, speech recognition, optical character reading, and image synthesis [7].



Figure 4: Photos on the use of virtual reality and artificial intelligence in research [7].

II.4. UAV-Based Smart Farming (Overview)

Unmanned Aerial Vehicles (UAVs), commonly known as drones, are increasingly becoming vital tools in smart farming a component of precision agriculture that leverages advanced technologies to optimize field-level management. UAV-based smart farming involves using aerial platforms equipped with sensors and cameras to collect high-resolution spatial data, enabling real-time monitoring, mapping, and analysis of agricultural environments.

These drones contribute significantly to improving crop yield, reducing resource wastage, and enhancing decision-making in farming operations. From seeding and spraying to health assessment and irrigation management, UAVs serve as versatile instruments in both large-scale and smallholder agriculture [14].

II.4.1. Implementation of UAVs in Agriculture

II.4.1.1. Crop Monitoring & Health Assessment

- UAVs equipped with multispectral or hyperspectral cameras help detect early signs of stress, disease, or pest infestation by analyzing vegetation indices like NDVI (Normalized Difference Vegetation Index) [15].

- Enables precision intervention, reducing pesticide usage and increasing plant health [15] .

II.4.1.2. Soil and Field Analysis

- UAVs assist in generating 3D maps of terrain, moisture levels, and soil texture [16] .
- Helps in pre-planting decision-making and assessing soil variability across fields [16].

II.4.1.3. Precision Spraying

- Drones with autonomous spraying systems can apply fertilizers, herbicides, and pesticides with high precision. [17] .
- Reduces chemical usage and exposure risks for human workers [17] .

II.4.1.4. Irrigation Management

- Thermal imaging cameras detect areas with water stress, allowing farmers to optimize irrigation schedules and reduce water waste [18] .

II.4.1.5. Crop Counting and Yield Prediction

- Using computer vision and AI models, UAVs can count plants/fruits and estimate expected yields, which supports supply chain planning and harvesting logistics [19] .

II.4.1.6. Livestock Monitoring

- Some UAVs are used for herding and tracking livestock, especially in large-scale ranching [20] .

II.4.2. Problems and Challenges of UAVs in Agriculture

Despite their potential, UAV-based farming faces several technical, economic, and regulatory challenges:

II.4.2.1. High Initial Cost

- The acquisition and maintenance costs of UAVs, especially those with advanced sensors (e.g., multispectral, LiDAR), are often prohibitive for smallholder farmers [21] .
- Software and data analysis tools add to the overall expense [21] .

II.4.2.2. Technical Complexity

- Requires specialized knowledge to operate UAVs, process data, and interpret outputs [22] .
- Not all farmers are trained in GIS, remote sensing, or AI-based analysis, necessitating external support [22] .

II.4.2.3. Limited Flight Time and Range

- Most commercial UAVs have short battery life (20–40 minutes), limiting their application over large fields without multiple recharges or battery swaps [23] .

II.4.2.4. Data Processing Challenges

- UAVs generate large volumes of data (images, video, thermal maps), which require powerful computing systems and often cloud-based platforms to process effectively [24] .

II.4.2.5. Regulatory Restrictions

- Many countries have strict regulations on drone flights (e.g., altitude limits, no-fly zones, licensing) [25] .
- Weather conditions, such as wind and rain, can also severely affect drone operations [25] .

II.4.2.6. Privacy and Security Concerns

- UAVs raise data privacy issues when flying over shared or neighboring land [26] .
- Cybersecurity risks also exist if drones are hacked or intercepted [26] .

II.4.2.7. Environmental and Safety Risks

- Risk of accidental crashes, especially in adverse weather or around power lines [27] .
- Improper handling or malfunctions can cause harm to people, animals, or crops [27] .

II.5. CNN

CNN (Convolutional Neural Network) is a deep learning algorithm which takes an image as an input and extracts all possible features from the images making the need for hand-engineered feature extraction obsolete. Compared to other algorithms, the pre-processing complexity of CNN is much less. In the earlier algorithms, filters were manually provided, but with server iterations of training, CNN has the capacity to learn those filters by themselves. The CNN was designed according to the circuit of neurons in the human brain.

The role of CNN is to compress the images in way that reduces processing complexities so that no features are lost and maximum accuracy is achieved. This characteristic is essential when we want to create a model does not only adept at extracting features but can also adapt to large datasets. The pooling layer has the task of decreasing the spatial area of the convolved features.

This serves the purpose of decreasing the effort to minimize the dimension of the matrix still maintaining matrix integrity. Max pooling and average pooling are the two types of pooling available. Max pooling gives the maximum value that the kernel encloses on the part of the image. On the other hand, average pooling gives the average of every value that the kernel encloses. High-level features are learned from the fully connected layer.

Then the images are converted to a column to its reduced form. These variables define the prominent features using provided numbers of iterations using SoftMax.

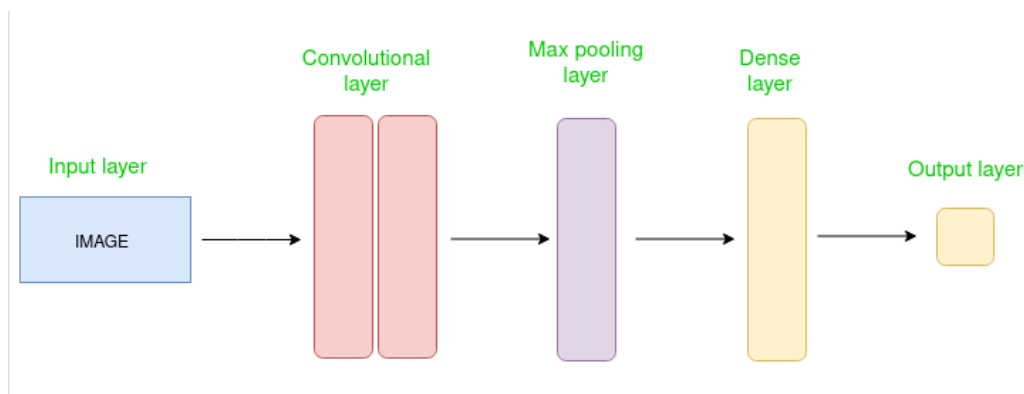


Figure 5: simple Architecture of CNN [5]

II.6. Image Processing

II.6.1. Definition of Image Processing

Image processing is a field positioned at the intersection of computer science, computer vision, and digital photography. It involves various operations and manipulations aimed at enhancing, analyzing, or extracting useful information from digital images with the purpose of:

- **Enhancing image quality:** This includes color correction, brightness and contrast adjustment, noise reduction, smoothing, and edge enhancement.

- **Restoring degraded images:** Techniques used to correct defects caused during image capture, such as blurring, scratches, and other forms of degradation.
- **Segmenting and analyzing:** Dividing an image into different parts or objects for deeper analysis. This may involve edge detection, feature extraction, and object recognition.
- **Compression and storage:** Reducing image file size to save storage space and facilitate transmission. Techniques include both lossless and lossy compression methods.
- **Image synthesis and generation:** Creating new images from data, often used in augmented reality, video games, and 3D modeling.

II.7. Transfer Learning

A unique event can be perceived in numerous deep neural networks: in the starting layers of the network, a deep learning model tries to learn a low level of features, like detecting edges, colors, variations of intensities, etc.

These features do not seem to fall under a specific dataset or an errand because of no matter what type of images we are processing either for detecting a lion or cars. In both cases, we must detect these low-level features. All these features occur regardless of the exact cost function or image dataset. Thus, learning these features in one task of detecting lion can be used in other tasks like detecting humans. This is what transfer learning is. Nowadays, it is very hard to see people training a complete CNN from the ground up, and it is common to use any pretrained model trained on a plethora of images in a similar task, e.g. models trained on ImageNet (1000 categories of around 1 million images), and use features from them to solve a new task .

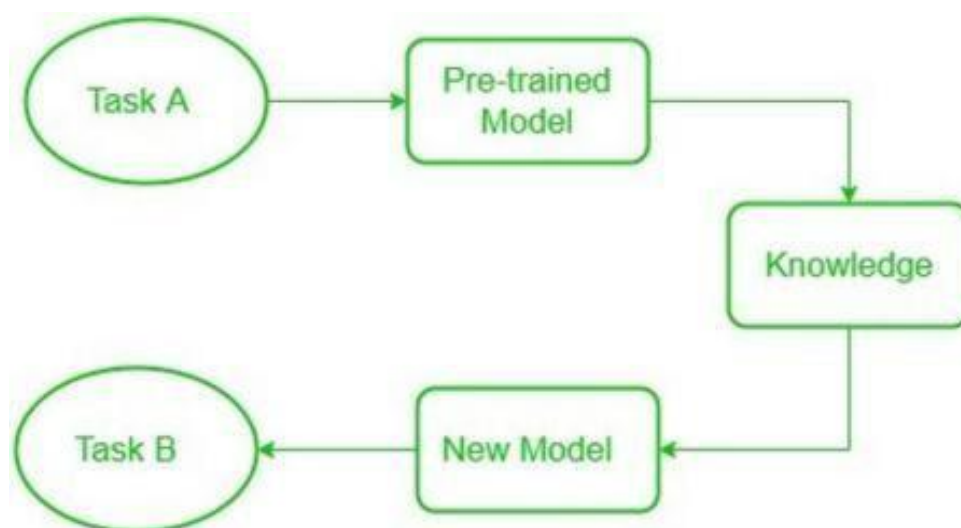


Figure 6: Block Diagram of transfer-learning [6]

II.8. Machine learning

Machine learning (ML) is a transformative subfield of artificial intelligence that empowers systems to automatically learn and improve from experience without being explicitly programmed. At its core, machine learning relies on algorithms that build mathematical models based on input data to make predictions or decisions. The architectural backbone of machine learning involves several key components: data preprocessing, feature extraction, model selection, training, and evaluation. The foundational architecture was laid by pioneers such as Arthur Samuel, who first coined the term in the 1950s, and was later formalized by researchers like Tom M. Mitchell, who defined a machine learning system as one that improves its performance on a task through experience. Architecturally, ML systems are structured in layers where data flows from raw input to predictive output, often through complex pipelines involving neural networks, decision trees, support vector machines, and ensemble methods. The architecture evolves depending on the problem supervised, unsupervised, or reinforcement learning and is enhanced by feedback loops, optimization strategies like gradient descent, and loss functions guiding the learning process. Modern advancements in computational power and big data have enabled deep learning architectures, such as convolutional and recurrent neural networks, to solve complex tasks in vision, language, and decision-making, continuing the legacy of structured, layered learning systems envisioned by early architects of the field.

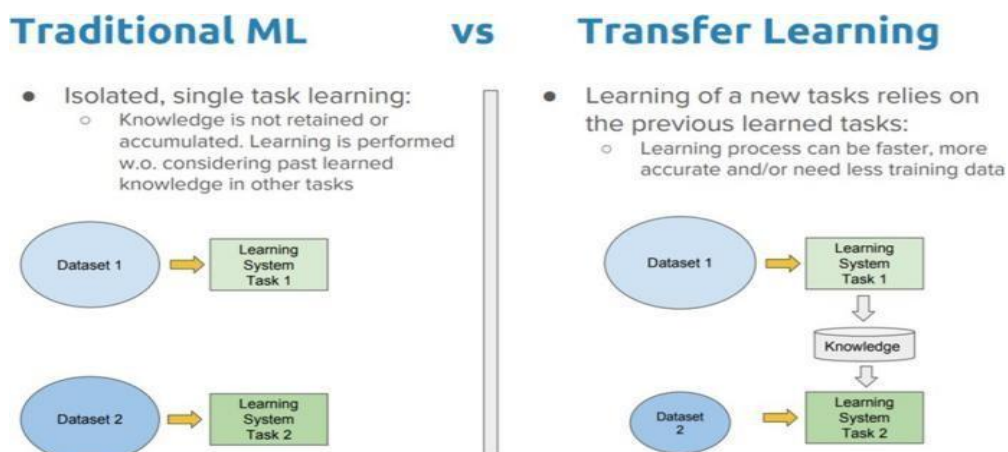


Figure 7: Traditional Learning vs Transfer Learning [9]

II.9. VGG16

The VGG-16 model is a convolutional neural network (CNN) architecture that was proposed by the Visual Geometry Group (VGG) at the University of Oxford. It is characterized by its depth, consisting of 16 layers, including 13 convolutional layers and 3 fully connected layers. VGG-16 is renowned for its simplicity and effectiveness, as well as its ability to achieve strong performance on various computer vision tasks, including image classification and object recognition. The model's architecture features a stack of convolutional layers followed by max-pooling layers, with progressively increasing depth. This design enables the model to learn intricate hierarchical representations of visual features, leading to robust and accurate predictions. Despite its simplicity compared to more recent architectures, VGG-16 remains a popular choice for many deep learning applications due to its versatility and excellent performance. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is an annual competition in computer vision where teams tackle tasks including object localization and image classification. VGG16, proposed by Karen Simonyan and Andrew Zisserman in 2014, achieved top ranks in both tasks, detecting objects from 200 classes and classifying images into 1000 categories.

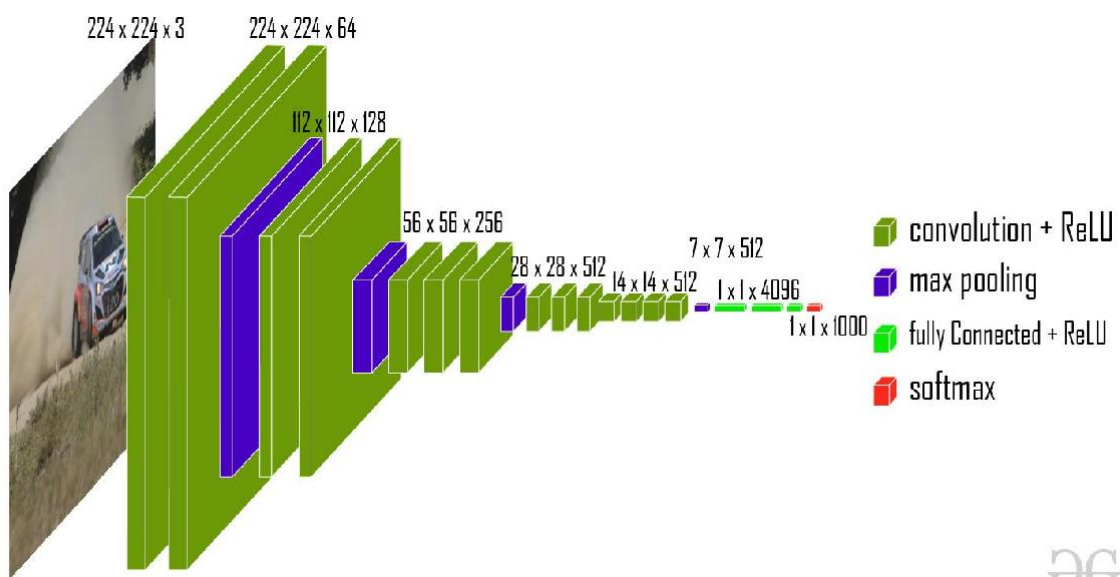


Figure 8: VGG-16 architecture [12]

II.10. VGG Architecture

VGG16 is a deep convolutional neural network architecture developed by Karen Simonyan and Andrew Zisserman from the Visual Geometry Group at the University of Oxford. It was introduced in 2014 through the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" and became notable for its simplicity, uniform design, and strong performance in the ImageNet competition. The architecture consists of 16 weight layers: 13 convolutional layers and 3 fully connected layers. It processes input images of size 224×224 with 3 color channels (RGB) and uses only small 3×3 convolution filters throughout the network, combined with 2×2 max pooling layers for down sampling. The model is structured in blocks: the first two blocks have two convolutional layers with 64 and 128 filters respectively, followed by max pooling; the next three blocks each have three convolutional layers with 256, 512, and 512 filters respectively, all followed by max pooling. After the convolutional part, the network has three dense (fully connected) layers: the first two with 4096 neurons and the third with 1000 neurons corresponding to the 1000 classes of ImageNet, followed by a SoftMax activation. Despite its relatively simple design, VGG16 has over 138 million parameters, making it computationally heavy and memory-intensive, especially in the fully connected layers. However, its deep structure and consistency make it highly effective for feature extraction and transfer learning in various computer vision tasks.

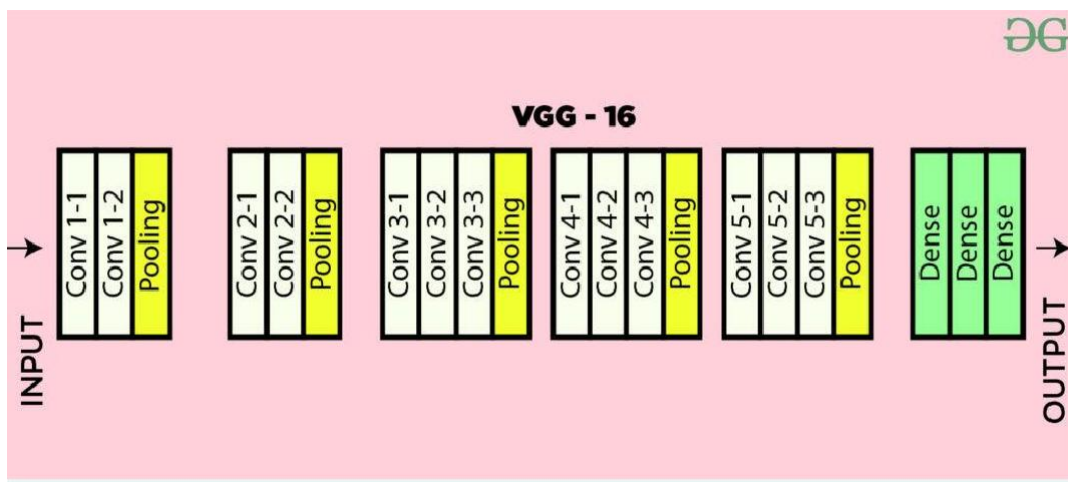


Figure 9: VGG-16 architecture Map [12]

II.10.1. The difference between the “VGG-16” and the “VGG-19” networks

Table 1: Comparison between VGG19 and VGG16 [12]

Layer	VGG16	VGG19
Size of Layer	41	47
Image Input Size	224x224 pixel	224x224 pixel
Convolutional Layer	13	16
Filter Size	64 & 128	64,128,256, & 512
ReLU	5	18
Max Pooling	5	5
FCL	3	3
Drop Out	0.5	0.5
Softmax	1	1

II.11. Keras function

Keras is a high-level neural networks API, written in Python and capable of running on top of deep learning frameworks like TensorFlow, Theano, or CNTK. It simplifies the creation, training, and deployment of deep learning models by offering an intuitive interface for both beginners and experts. Keras operates through a functional architecture built around modular blocks including models, layers, optimizers, loss functions, metrics, callbacks, and preprocessing utilities.

The core component is the Model API, which includes the Sequential model—ideal for linear layer stacks—and the Functional API, allowing the construction of more complex, non-linear architectures like multi-input or multi-output models, shared layers, and directed acyclic graphs. Layers like Dense, Conv2D, LSTM, and Dropout represent neural components that can be stacked or combined flexibly, while optimizers such as SGD, Adam, and RMSprop adjust learning based on gradients to minimize a loss function like categorical_crossentropy or mean_squared_error. Keras also includes metrics (e.g., accuracy, precision) to monitor performance during training and validation, and callbacks such as EarlyStopping, ModelCheckpoint, and TensorBoard for monitoring and controlling the training loop dynamically. For data handling, Keras provides preprocessing functions to load, augment, and batch datasets efficiently (e.g., ImageDataGenerator, Tokenizer, pad_sequences).

Architecturally, Keras is designed with modularity, extensibility, and minimalism at its core: every element is a standalone, configurable object that can be plugged into a model seamlessly. This design philosophy allows Keras to act as a rapid prototyping tool while remaining powerful enough for production-level deployment. It abstracts away much of the complexity of deep learning backend frameworks, enabling users to focus on building and testing models without getting lost in low-level operations. [11]

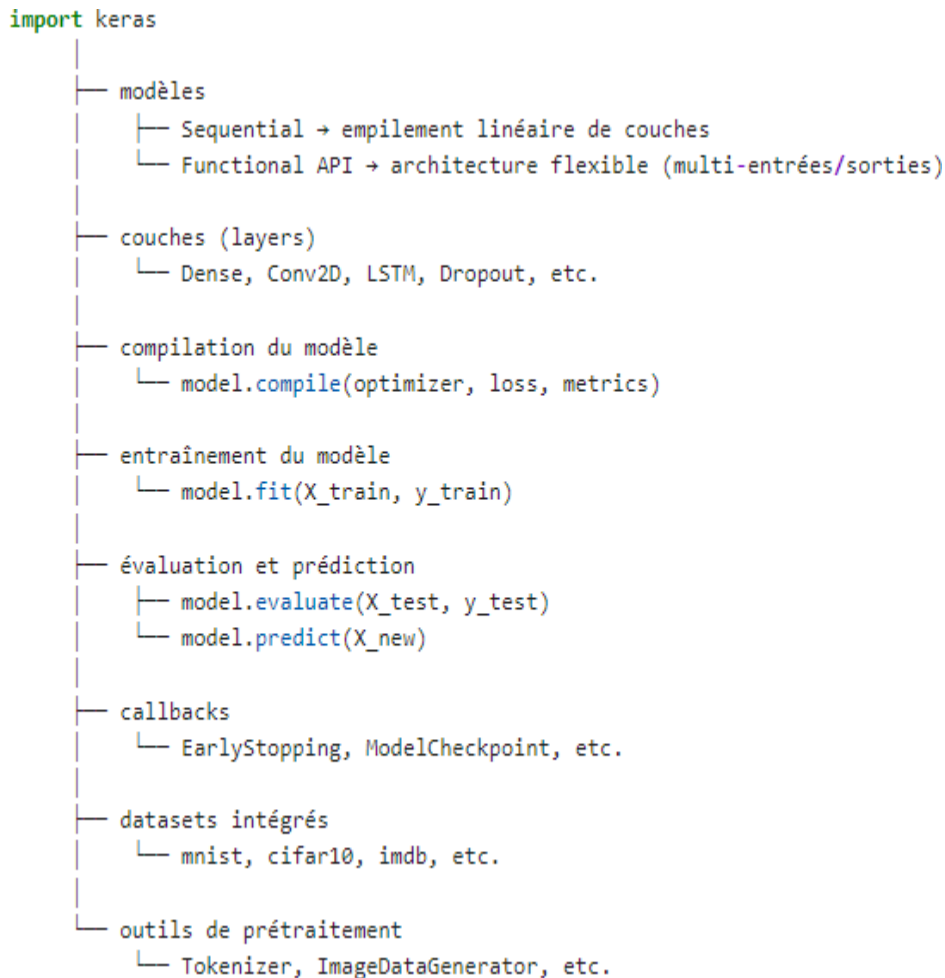


Figure 10: Summary Diagram of the Main Functions Through Keras

II.12. TensorFlow function

TensorFlow is an open-source end-to-end machine learning framework developed by Google, designed for building and deploying machine learning and deep learning models at scale. It operates using a highly flexible, layered architecture that supports development from simple prototypes to complex production systems. At its core, TensorFlow uses computational graphs, where nodes represent operations (like addition or matrix multiplication) and edges represent tensors (multidimensional arrays of data).

This graph-based architecture enables optimized execution across different platforms (CPUs, GPUs, TPUs) and environments (mobile, edge, cloud). The foundational API provides low-level operations for tensor manipulation (`tf.constant`, `tf.Variable`, `tf.matmul`, etc.), allowing fine-grained control for researchers and developers.

On top of this, TensorFlow offers high-level APIs such as `tf.keras`, which simplifies model building, training, evaluation, and deployment through familiar objects like `Sequential`, `Model`, `Layer`, `compile()`, and `fit()`. TensorFlow includes powerful tools for automatic differentiation (`tf.GradientTape`) essential for training deep networks, and supports various optimizers (SGD, Adam, Adagrad) and loss functions (`categorical_crossentropy`, `mse`) for different learning tasks. It also provides advanced modules for data pipeline management using the `tf.data` API, which allows efficient loading, preprocessing, batching, and shuffling of large datasets. For deployment, TensorFlow supports TensorFlow Lite (for mobile and embedded devices), TensorFlow.js (for browser and JavaScript apps), and TensorFlow Serving (for scalable inference in production). The architecture is extensible, with support for custom layers, training loops, distributed training (`tf.distribute.Strategy`), and integration with tools like TensorBoard for visualization and TF Hub for model reuse. Designed with scalability, flexibility, and performance in mind, TensorFlow bridges the gap between research and production, enabling developers to move seamlessly from experimentation to deployment in real-world environments.

II.13. ImageDataGenerator

`ImageDataGenerator` is a high-level utility provided by the `tf.keras.preprocessing.image` module in TensorFlow that plays a crucial role in preparing image data for training deep learning models, especially in computer vision tasks. It is designed to handle image loading, preprocessing, augmentation, and real-time data feeding in a memory-efficient and automated manner. The architecture of `ImageDataGenerator` revolves around a generator-based pipeline that yields batches of preprocessed images directly to the training loop, which is ideal for handling large datasets that cannot fit entirely into memory. It includes functionality for rescaling pixel values (e.g., `rescale=1./255` to normalize images), and powerful data augmentation techniques such as rotation, shifting, zooming, flipping, shearing, and brightness adjustment, which help improve model generalization by artificially expanding the training set with realistic image variations.

The `flow_from_directory()` method is a key component that loads images from a directory structure organized by class labels, automatically labeling them and generating batches for training, validation, or testing. Alternatively, `flow()` supports direct input from Numpy arrays, and `flow_from_dataframe()` enables integration with data in tabular formats.

Architecturally, `ImageDataGenerator` sits between raw data and the model, operating in real-time without needing to store augmented copies of data on disk. This streamlines the training process, especially for convolutional neural networks (CNNs), by continuously feeding fresh, varied data that reduces overfitting and enhances robustness. Furthermore, it supports shuffling, batch sizing, and label encoding as part of the pipeline, making it a comprehensive tool for image preprocessing and augmentation. Although TensorFlow 2.x has moved toward using the more flexible `tf.data` API for production pipelines, `ImageDataGenerator` remains widely used for its simplicity and effectiveness in prototyping and small-to-medium scale computer vision tasks. Its modular, plug-and-play architecture integrates directly with `model.fit()` or `model.fit_generator()` methods, allowing users to efficiently train models on enriched image datasets with minimal code and maximal flexibility.

II.14. Flatten and dense

Flatten and Dense are fundamental layers in the architecture of neural networks, especially within the Keras and TensorFlow frameworks, and they play a central role in connecting different parts of a model. The Flatten layer acts as a structural bridge between convolutional or recurrent layers and fully connected (dense) layers; it reshapes a multi-dimensional tensor (e.g., a 2D feature map from a convolutional layer) into a one-dimensional vector while preserving the batch size. This transformation is essential because fully connected layers require 1D input vectors. Flatten performs no learning—its role is purely structural—but it enables the transition from spatial feature extraction to decision-making layers. In contrast, the Dense layer is a fully connected neural layer, where every input neuron is connected to every output neuron. Architecturally, a Dense layer performs a linear transformation ($\text{output} = \text{activation}(Wx + b)$) where W is the weight matrix, x is the input, b is the bias vector, and the result is passed through an activation function such as ReLU, sigmoid, or softmax. This layer learns patterns and decision boundaries based on input features and is commonly used in both intermediate and output layers of neural networks.

Dense layers are critical for classification, regression, and decision-making tasks as they combine features extracted by earlier layers and map them to output labels or predictions. Both Flatten and Dense integrate seamlessly within the Sequential or Functional API of Keras/TensorFlow, and their architecture supports custom configurations such as specifying the number of units, activation functions, kernel initializers, and regularization strategies.

Together, Flatten and Dense form the backbone of the transition from raw feature extraction to high-level reasoning in deep learning pipelines, making them indispensable components in neural network design.

II.15. Earlystopping

EarlyStopping is a crucial regularization technique in deep learning that helps prevent overfitting by halting the training process when a monitored performance metric, such as validation loss or accuracy, stops improving. It is implemented as a callback in Keras and is integrated directly into the training loop via the `Model.fit()` function. Architecturally, EarlyStopping is built as a class that inherits from `tf.keras.callbacks.Callback`, and it overrides key methods like `on_train_begin`, `on_epoch_end`, and `on_train_end`. At the start of training, it initializes internal variables to track progress. After each epoch, it evaluates whether the monitored metric has improved; if not, it increases a patience counter. If the counter exceeds a predefined threshold (set via the `patience` parameter), training stops. If the `restore_best_weights` flag is set to `True`, it rolls back the model to the best-performing weights recorded during training. The callback can operate in different modes such as `'min'` or `'max'`, depending on whether the metric is expected to decrease (like loss) or increase (like accuracy). This mechanism is particularly useful as it not only reduces training time and computational cost but also ensures that the model generalizes better by avoiding overfitting to the training data. EarlyStopping acts as an intelligent checkpoint system that guards model performance dynamically, making it essential for robust model development.

II.16. Matplotlib.pyplot

Matplotlib.pyplot is a widely used plotting module in Python, part of the larger matplotlib library, and is primarily designed for creating static, interactive, and animated visualizations. It is often imported using the alias `plt` with the command `import matplotlib.pyplot as plt`. Architecturally, pyplot functions as a state-machine-based interface built on top of matplotlib's object-oriented (OO) API. It mimics the MATLAB plotting style, making it easy for users to generate plots with minimal code. Behind the scenes, pyplot manages the creation and manipulation of figure and axes objects automatically, allowing users to focus on plot content rather than the underlying figure hierarchy.

For instance, when a function like `plt.plot()` is called, pyplot checks if a figure or axes exist, creates them if not, and plots the data accordingly.

This abstraction is ideal for quick data visualization and prototyping. Internally, pyplot uses a rendering engine to draw the plots on various backends (e.g., Agg for PNGs, TkAgg for GUIs). It provides a vast collection of functions to control every aspect of a plot, including titles, labels, ticks, legends, grids, and layout. matplotlib.pyplot supports various plot types such as line charts, bar charts, histograms, scatter plots, and more. It is particularly powerful when combined with data libraries like NumPy or Pandas, enabling high-quality and highly customizable visual representations of data for analysis and presentation. [13]

II.17. MobileNet

MobileNet is a class of lightweight deep convolutional neural network architectures designed specifically for mobile and embedded vision applications, where computational resources and memory are limited. Architecturally, MobileNet is built on a highly efficient model design based on depthwise separable convolutions, which significantly reduce the number of parameters and computational cost compared to traditional convolutional networks like VGG or ResNet. Instead of using a single standard convolution, MobileNet factorizes it into two layers: a depthwise convolution, which applies a single filter per input channel, followed by a pointwise convolution (a 1×1 convolution), which combines the outputs of the depthwise convolution.

This architectural innovation drastically lowers the number of multiply-add operations (FLOPs) while preserving most of the accuracy, making it ideal for real-time inference on devices like smartphones, IoT hardware, and edge computing platforms. There are multiple versions of MobileNet: MobileNetV1, V2, and V3, each introducing additional improvements. MobileNetV2 introduces an inverted residual structure and linear bottlenecks, which enhance feature reuse and reduce information loss. MobileNetV3, the latest in the family, combines ideas from V2 with Neural Architecture Search (NAS) to automatically discover the most efficient architectures for a given hardware constraint, and includes additional components like SE blocks (Squeeze-and-Excitation) for better feature recalibration. In TensorFlow or Keras, MobileNet can be easily imported via `from keras.applications import MobileNet` or using MobileNetV2 and MobileNetV3 variants.

These models are widely used in transfer learning for tasks like image classification, object detection, and segmentation by leveraging their pre-trained weights on ImageNet and fine-tuning them for specific tasks. In summary, MobileNet achieves an optimal balance between speed and accuracy, making it a cornerstone model for deploying deep learning on edge devices.

II.18. DenseNet 201

DenseNet-201 is a deep convolutional neural network that belongs to the DenseNet (Densely Connected Convolutional Networks) family, specifically designed to improve the flow of information and gradients throughout the network. Its architecture is based on the principle of dense connectivity, where each layer receives the feature maps of all preceding layers as input, and passes its own feature maps to all subsequent layers. This results in 201 total layers, making it a very deep model while still being parameter-efficient. Architecturally, DenseNet-201 is composed of multiple dense blocks, each containing a sequence of convolutional layers, and separated by transition layers that use 1x1 convolutions and pooling to compress feature maps and control model complexity. Each layer within a dense block consists of a batch normalization layer, a ReLU activation, a 1x1 bottleneck convolution, followed by a 3x3 convolution, with the output concatenated to the inputs of the next layer—creating a rich network of feature reuse and preventing vanishing gradients. DenseNet-201 uses a growth rate (usually 32) to control how many new feature maps each layer contributes. Despite its depth, the dense connectivity pattern significantly reduces the number of parameters compared to traditional architectures like VGG or ResNet, and it promotes better feature propagation and regularization. In practice, DenseNet-201 is used for high-level vision tasks such as image classification and medical imaging, often pretrained on large datasets like ImageNet and fine-tuned for specific tasks. The combination of deep hierarchical features and efficient connections makes DenseNet-201 both powerful and computationally feasible for complex real-world applications.

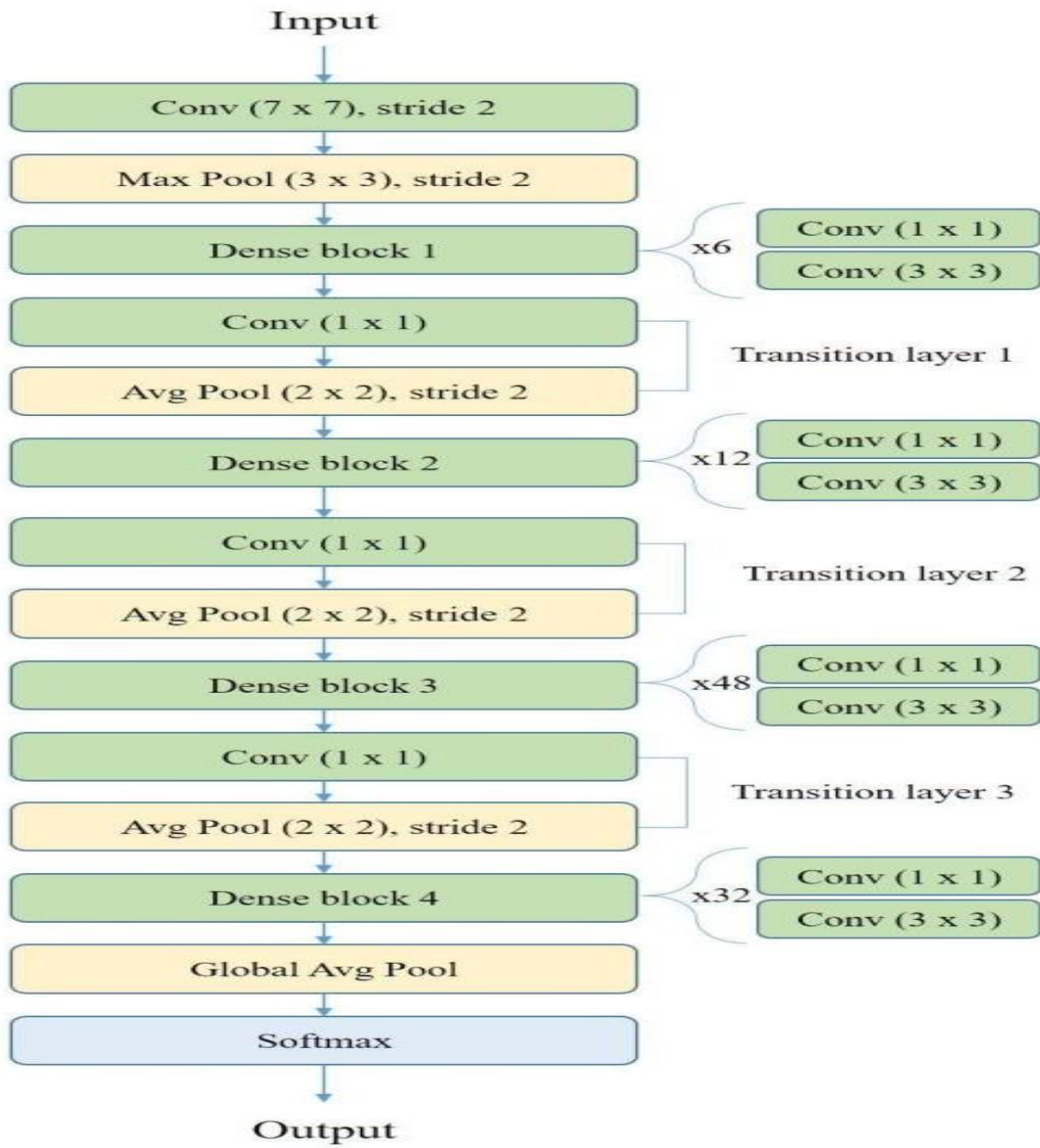


Figure 11: DenseNet 201 Architecture

II.19. Load model

In the context of deep learning, especially when using TensorFlow and Keras, the `load_model` function plays a crucial role in the deployment and reuse of trained models. This function, available via `from tensorflow.keras.models import load_model`, allows you to load a previously saved model, including its architecture, weights, optimizer configuration, and training state. The architecture of this function was designed to support reproducibility and portability of models across different environments and applications. Under the hood, `load_model` works by reading a file typically in the HDF5 (.h5) format or the TensorFlow SavedModel format that stores all components of the model. When invoked, it reconstructs the exact same model object used during training, enabling further training (fine-tuning), evaluation, or inference without redefining the architecture or reloading the weights manually. The modular architecture of `load_model` ensures compatibility with custom objects through the `custom_objects` argument, which is essential for models that use custom layers, loss functions, or metrics. This functionality is part of the broader architectural philosophy in TensorFlow/Keras to streamline the machine learning lifecycle, from model definition and training to saving and loading for production or experimentation.

II.20. Sklearn

Scikit-learn, commonly imported as `sklearn`, is a powerful and widely-used machine learning library in Python, designed with a clean and consistent architecture for building and evaluating ML models. Developed initially by David Cournapeau as part of the Google Summer of Code project, and later maintained by a large community of contributors, scikit-learn is built on top of foundational scientific libraries like NumPy, SciPy, and matplotlib. Its architecture is modular and object-oriented, which allows for easy experimentation and interchangeability of components such as classifiers, regressors, and preprocessing tools. At its core, scikit-learn provides a unified interface for supervised and unsupervised learning tasks such as classification, regression, clustering, dimensionality reduction, and model selection through simple and intuitive APIs. The central design pattern of `fit()`, `predict()`, and `transform()` methods ensures consistency across different models and transformers.

For example, whether you're using a Support Vector Machine (SVC), a Decision Tree (DecisionTreeClassifier), or a preprocessing scaler (StandardScaler), you can apply the same method calls, which simplifies the machine learning workflow. Scikit-learn also incorporates tools for model evaluation (e.g., cross-validation, confusion matrices, classification reports), hyperparameter tuning (e.g., GridSearchCV), and pipelines (Pipeline objects), allowing for efficient and reproducible end-to-end model development. Its architecture reflects a balance between simplicity and flexibility, making it ideal for both beginners and researchers who want robust, fast, and scalable ML implementations. [12]

II.21. Confusion matrix

The `confusion_matrix` function in scikit-learn is a vital evaluation tool used to measure the performance of classification models by comparing predicted and actual class labels. Architecturally, it is designed to provide a structured summary of prediction results in the form of a square matrix, where each row represents the instances of an actual class, and each column represents the instances of a predicted class. This function is accessed via `from sklearn.metrics import confusion_matrix`, and it accepts two primary arguments: the true labels (`y_true`) and the predicted labels (`y_pred`). Under the hood, `confusion_matrix` constructs the matrix by counting the number of correct and incorrect predictions for each class. In binary classification, it shows true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), which are foundational for computing performance metrics such as accuracy, precision, recall, and F1-score. In multiclass problems, the matrix expands to reflect all possible pairwise comparisons between true and predicted classes, helping identify patterns like class imbalance or frequent misclassification. From an architectural perspective, `confusion_matrix` integrates seamlessly with other scikit-learn metrics and visualization tools, and can be customized with parameters such as `labels` to order or limit the output matrix, and `normalize` to return proportions instead of raw counts.

This modular and interpretable design makes it a core component in diagnosing and improving classification models in real-world machine learning pipelines.

II.22. Conclusion

In this chapter, we explored the integration of artificial intelligence and image processing in the context of citrus disease detection and management. We discussed the pressing challenges in the citrus industry due to diseases such as citrus greening, black spot, and canker, and how traditional detection methods are often insufficient for timely and large-scale interventions. The application of machine learning and deep learning—particularly Convolutional Neural Networks (CNNs)—was shown to offer powerful solutions for automating disease detection with high accuracy. We examined key technologies including transfer learning, object detection algorithms, and drone-based monitoring, as well as core tools and libraries like TensorFlow, Keras, NumPy, and Pandas that support this work. Finally, we reviewed architectural models like VGG, DenseNet, and MobileNet, which play a crucial role in enhancing classification performance.

This technological foundation sets the stage for the next chapter, which will focus on the methodology and experimental setup used to implement and evaluate AI-based citrus disease detection systems in practice.

Chapter III : Simulation and results

III.1. Introduction

To address these challenges, the integration of Artificial Intelligence (AI) particularly Machine Learning (ML) and Deep Learning (DL) with image processing techniques is emerging as a powerful solution for automated disease detection. By leveraging large datasets of citrus images, AI models can be trained to identify subtle patterns in leaves, fruits, or branches that may be invisible to the human eye. Image processing methods such as color space transformation (e.g., RGB to HSV), segmentation, edge detection, and morphological filtering allow for preprocessing and feature extraction, improving the accuracy of classification. In this context, Convolutional Neural Networks (CNNs) have become the backbone of image-based plant disease classification.

III.2. Platform used for implementation

III.2.1. Software

III.2.1.1. Python 3.11.3

Python is an object-oriented, open-source programming language with dynamic semantics. Its syntax is simple and readable, making it easy to learn and maintain programs. The Python interpreter and its extensive standard library are freely available in source form for all major platforms. [9]



III.2.1.2. Jupyter 6.5.4

Jupyter Notebook was released in 2015 as an interactive environment for running code directly in the browser. It is a notebook-creation application developed under the Jupyter Project. Jupyter Notebook offers fast and interactive ways to prototype and explain code, as well as to explore and visualize data. It facilitates the integration of code, text, and images in a seamless and user-friendly manner. [10]

III.3. Working Principal

It is a systematic process where a computer learns to identify different conditions of citrus fruits (healthy or diseased) by looking at many examples.

III.3.1. Data Input (Gathering the Examples)

III.3.1.1. Principle

The algorithm begins by collecting a large collection of images of citrus fruits (oranges). These images are carefully labeled to indicate their condition: "fresh" (healthy), "canker," "black spot," or "greening."

III.3.1.2. Purpose

This labeled dataset serves as the "knowledge base" for the computer. It's like showing someone many examples of apples and oranges and telling them which is which, so they can learn to differentiate.

III.3.2. Preprocessing (Getting Ready to Learn)

III.3.2.1. Principle

Raw images from cameras can vary widely in size, lighting, and orientation. This stage standardizes them so the computer can process them efficiently.

III.3.2.2. Purpose

Images are typically resized to a uniform dimension (e.g., 128x128 pixels), their colors might be normalized, and sometimes "augmented" (slightly rotated, zoomed, or flipped) to create even more variations from the existing images. This makes the learning process more robust and prevents the model from being too specific to the exact images it saw during training.

III.3.3. Model Training (The Learning Phase)

III.3.3.1. Principle

This is where the core "learning" happens, often using a type of Artificial Intelligence called a *Convolutional Neural Network (CNN)*. The preprocessed images and their labels are fed into this network.

III.3.3.2. Process

The CNN analyzes patterns and features within the images (e.g., the texture of a black spot, the color changes of greening, or the smooth surface of a fresh orange). Through many iterations, the network adjusts its internal parameters to best associate specific visual patterns with their corresponding disease labels. It continuously tries to minimize errors in its predictions.

III.3.3.3. Outcome

After training, the model becomes capable of recognizing these visual patterns and making predictions based on them.

III.3.4. Output Prediction (Identifying New Cases)

III.3.4.1. Principle

Once the model is trained, it's ready to be used on *new*, unseen orange images to predict their condition.

III.3.4.2. Process

You provide a new orange image (e.g., from a farm or a market). This image goes through the same preprocessing steps as the training data. The preprocessed image is then fed into the now-trained CNN. The CNN analyzes the image and outputs a probability for each possible condition (canker, black spot, greening, fresh).

III.3.4.3. Result

The condition with the highest probability is given as the model's prediction, telling us whether the orange is fresh or affected by a specific disease.

In essence, the algorithm learns from a vast collection of examples to become an expert at visually diagnosing orange fruit diseases, allowing for quick and automated inspection.

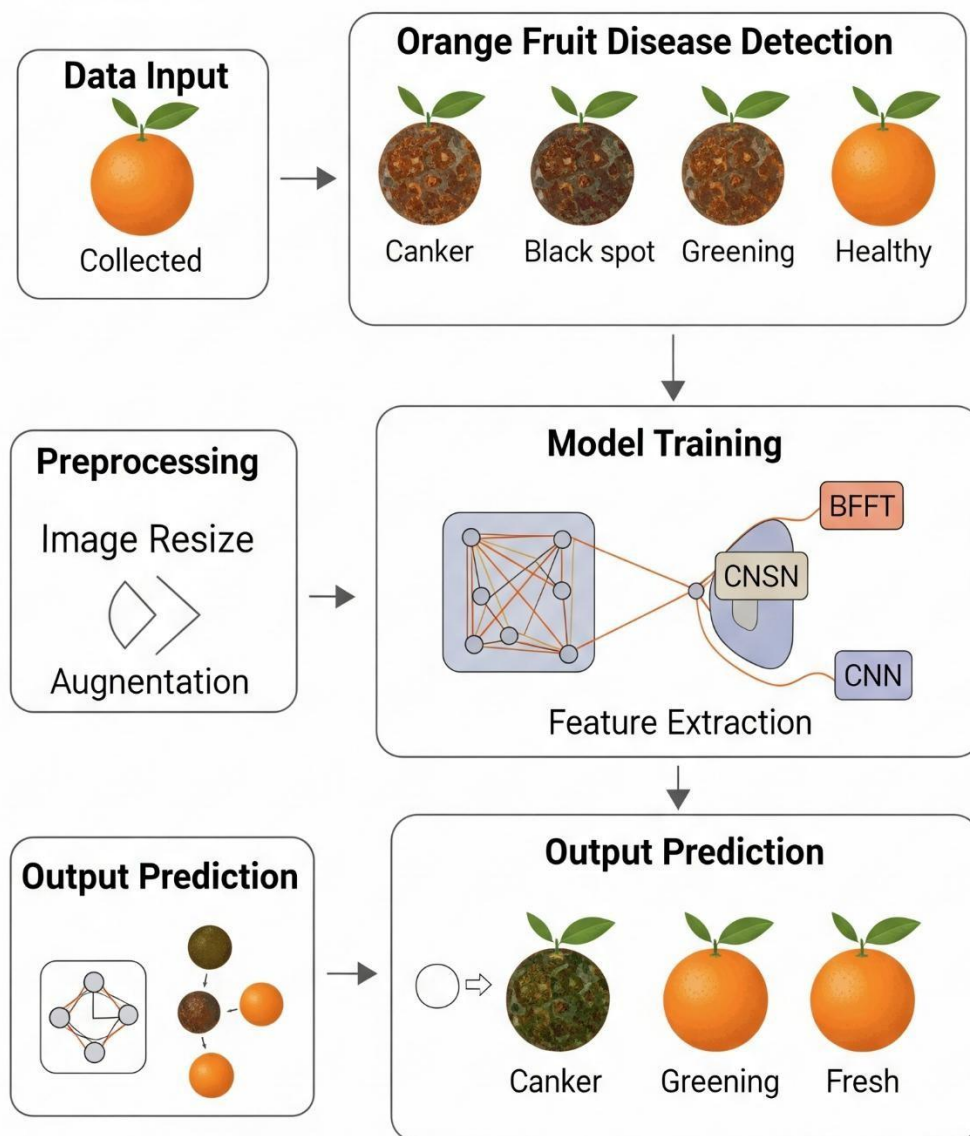


Figure 12: Functioning Diagrammed of the Machine (Model)

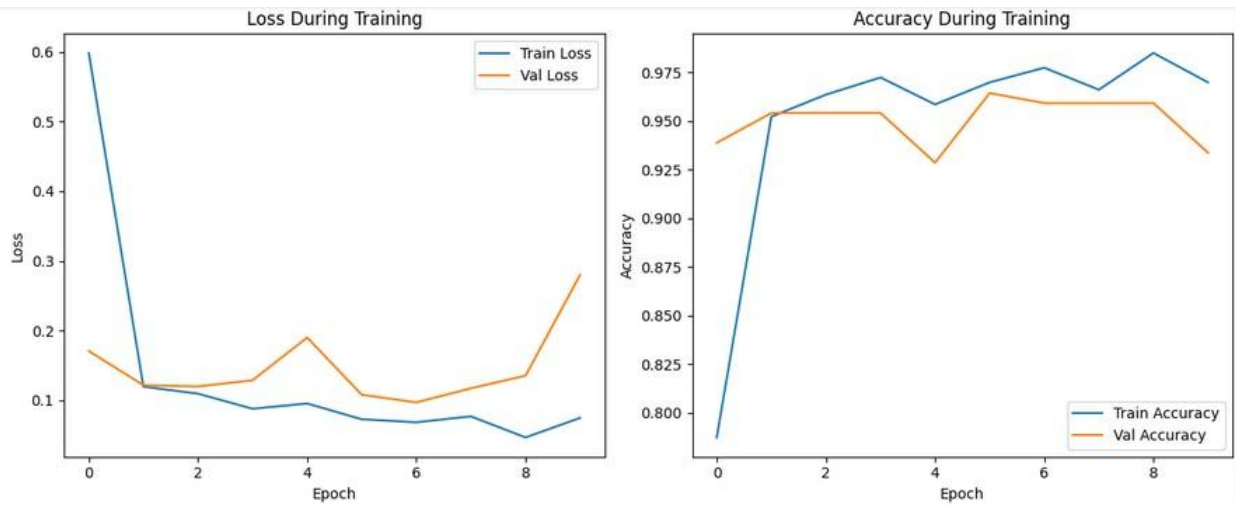
III.4. The LOSS and ACCURACY of the validation and the training during the training

III.4.1. Validation

Validation is the phase where the model is tested during training on new data it hasn't seen before to check how well it performs.

III.4.2. Train

Training is the phase where your model learns from labeled data.



III.4.2.1. Loss During Training

III.4.2.1.1. Train Loss (blue)

The training loss demonstrates a consistent downward trend across epochs, suggesting that the model is effectively learning the underlying patterns and structures within the training data. This progressive reduction in loss reflects the model's increasing ability to minimize prediction errors through weight adjustments during backpropagation. The absence of significant fluctuations or spikes in the training loss further indicates a stable learning process and convergence, implying that the learning rate is well-tuned and that the model is not experiencing instability during optimization. Moreover, the steady decline suggests that the model is not underfitting the data, as it is successfully capturing the complexity of the input without struggling to reduce the error. This behavior is characteristic of a model that is appropriately expressive and is benefiting from sufficient training data and well-calibrated hyperparameters.

III.4.2.1.2. Validation Loss (orange)

The validation loss initially decreases during the first few epochs (notably between epochs 0 and 1), indicating that the model begins to generalize effectively to unseen data early in the training process. After this initial drop, the loss remains relatively stable across subsequent epochs, with minimal fluctuations. This suggests that the model maintains a consistent level of performance on the validation set, neither improving significantly nor deteriorating. Such a plateau may reflect that the model has reached a point where additional training offers diminishing returns in terms of generalization. It also indicates that the model is not underfitting, as it was able to reduce validation loss initially, and is not showing signs of instability or erratic behavior during training.

III.4.2.2. Accuracy During Training


III.4.2.2.1. Train Accuracy (blue)

The training accuracy exhibits a smooth and continuous upward trajectory across the training epochs, ultimately reaching approximately 98%. This steady improvement indicates that the model is progressively enhancing its ability to correctly classify the training samples. The absence of abrupt jumps or volatility suggests that the optimization process is stable and that the learning rate is appropriately configured. Achieving high training accuracy reflects the model's strong capacity to capture and represent the underlying features of the training data, further supporting the observation that the model is not underfitting. Such performance may be attributed to an effective architecture, sufficient training data, and appropriate hyperparameter selection, all contributing to successful pattern recognition and classification within the training set.

III.4.2.2.2. Validation Accuracy (orange)

The validation accuracy starts at a relatively high level, approximately 94%, and remains consistently strong throughout the training process. This indicates that the model is capable of generalizing well to unseen data from the early stages of training. The sustained high performance on the validation set suggests that the learned features are not merely specific to the training data but are also representative of the broader data distribution. The lack of significant drops or instability in validation accuracy further implies that the model maintains its generalization capacity over time. This behavior reflects a well-balanced training process, where the model avoids both underfitting and severe overfitting during the observed epochs.

- **Results**

```
Test Loss: 0.1063
Test Accuracy: 0.9697
4/4  5s 979ms/step
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.91	0.95	22
1	0.88	1.00	0.94	22
2	1.00	0.97	0.98	33
3	1.00	1.00	1.00	22
accuracy			0.97	99
macro avg	0.97	0.97	0.97	99
weighted avg	0.97	0.97	0.97	99

Confusion Matrix:

```
[[20  2  0  0]
 [ 0 22  0  0]
 [ 0  1 32  0]
 [ 0  0  0 22]]
```

--- Predictions for Uploaded Images in /home/kali/Downloads/tests ---

III.5. Test

III.5.1. Test Loss

The model achieves a low-test loss of 0.1063, indicating a minimal error when making predictions on completely unseen data. This low value demonstrates that the model has successfully generalized beyond the training and validation datasets and is capable of maintaining strong performance in a real-world inference context. A small test loss, when accompanied by high accuracy, as observed here (96.97%), suggests that the model is neither overfitting nor underfitting, but rather has achieved an optimal balance between learning the training data and generalizing to new samples. The consistency between training, validation, and test metrics further reinforces the robustness and reliability of the model's predictive capability.

III.5.2. Test Accuracy

The model achieves a high-test accuracy of 96.97%, which is indicative of excellent generalization to previously unseen data. This performance reflects the model's ability to accurately classify new examples that were not part of the training or validation sets, suggesting that the learned features are robust and transferable. Such a high level of accuracy on the test set implies that the model is well-suited for deployment in practical, real-world scenarios where reliability and consistency are critical. The alignment between test accuracy and both training and validation accuracy further confirms that the model maintains strong predictive performance across all phases of evaluation without exhibiting signs of overfitting.

Table 2: Classification Report

Class	Precision	Recall	F1-score	Support
0 (e.g. Fresh)	1.00	0.91	0.95	22
1 (e.g. Blackspot)	0.88	1.00	0.94	22
2 (e.g. Canker)	1.00	0.97	0.98	33
3 (e.g. Greening)	1.00	1.00	1.00	22

III.5.3. Interpretation

III.5.3.1. Precision

Precision reflects the proportion of correct positive predictions among all instances predicted as positive. In this context, the model demonstrates consistently high precision across all classes, with values ranging from 0.88 to 1.00. Such results indicate that the majority of the predicted cases for each disease class are accurate, minimizing false positives. Notably, classes such as 'Fresh,' 'Canker,' and 'Greening' exhibit perfect precision (1.00), meaning the model made no incorrect positive predictions for these categories. The slightly lower precision for 'Blackspot' (0.88) still represents strong performance, though it suggests a few instances of misclassification. Overall, the high precision scores across all classes confirm the model's reliability in making confident and accurate predictions, which is essential in practical applications such as disease detection in agriculture, where false alarms could lead to unnecessary interventions.

III.5.3.2. Recall

Recall measures the proportion of actual positive cases that were correctly identified by the model, reflecting its ability to capture relevant instances. The recall scores in this evaluation are consistently high across all classes, ranging from 0.91 to 1.00, indicating that the model is highly effective in detecting true positives. Of particular note is Class 1 (e.g., Blackspot), which achieved a perfect recall of 1.00, signifying that all actual instances of this disease were successfully identified without any false negatives. Similarly, Classes 2 (Canker) and 3 (Greening) also show excellent recall (0.97 and 1.00, respectively), suggesting robust detection capability. While Class 0 (Fresh) exhibits a slightly lower recall of 0.91, it remains strong and suggests only a small number of misclassifications. Overall, these results confirm the model's high sensitivity, especially crucial in agricultural disease detection, where failing to identify diseased samples can lead to significant economic and ecological consequences.

III.5.3.3. F1-score

The F1-score, defined as the harmonic mean of precision and recall, provides a comprehensive measure of a model's performance by balancing its ability to avoid both false positives and false negatives. In the presented results, all F1-scores are remarkably high, with values exceeding 0.94 across all classes. This indicates that the model consistently maintains strong precision and recall simultaneously, reflecting its robustness in classifying each disease category. Notably, Class 3 (e.g., Greening) achieves a perfect F1-score of 1.00, while other classes such as Canker (0.98), Fresh (0.95), and Blackspot (0.94) also demonstrate excellent performance. These consistently high F1-scores confirm the model's reliability and suitability for tasks requiring both accurate detection and minimal misclassification, such as early identification of plant diseases in agricultural settings.

III.5.3.4. Macro Avg and Weighted Avg

Both the macro average and weighted average are reported at 0.97, indicating that the model maintains consistently high performance across all classes, even in the presence of slight class imbalance. This demonstrates that the model is not only accurate on the dominant classes but also performs well on less represented categories, reflecting balanced and robust classification behavior.

III.5.4. Confusion Matrix

```
[20  2  0  0]
[ 0 22  0  0]
[ 0  1 32  0]
[ 0  0  0 22]
```

III.5.4.1. Interpretation

- Diagonal values (20, 22, 32, 22) are **correct predictions**.
- Only 3 misclassifications total:
 - 2 samples from class 0 misclassified (possibly predicted as class 1).
 - 1 sample from class 2 misclassified (possibly as class 1).
- No confusion between completely unrelated classes.

III.5.5. Model Prediction Output

We have downloaded one image of the desired fruit (Orange), a fresh one and uploaded it to the model and we have let it decide whether it is a good orange or a bad one :

```
f (7).png
1/1 ————— 2s 2s/step
✔ Prediction: fresh (Healthy)
```

The model accurately identified the fruit as 'Fresh,' demonstrating its ability to correctly distinguish between healthy and diseased samples. This successful prediction provides practical evidence of the model's effectiveness and reliability in recognizing the health status of citrus fruits. It also reinforces the model's generalization capability beyond the training dataset, indicating its potential for deployment in real agricultural settings where automated fruit quality assessment is required.

III.6. Conclusion

This chapter presented a comprehensive overview of the advanced methodologies and deep learning techniques employed for the recognition of citrus diseases. The core of the proposed work involved the development and optimization of a Convolutional Neural Network (CNN) model tailored to accurately detect and classify various citrus leaf conditions, including healthy samples and multiple disease types. In addition to describing the model architecture and enhancements, we also provided an in-depth explanation of the tools, libraries, and computational frameworks used throughout the implementation. The chapter further outlined the characteristics and preparation of the dataset, including preprocessing strategies and data augmentation techniques to improve model robustness. A clear, step-by-step description of the training, validation, and testing pipeline was provided to ensure reproducibility and transparency. The results obtained confirm the effectiveness of the proposed model in distinguishing between citrus diseases with high accuracy, underscoring the potential of deep learning as a reliable solution in precision agriculture and automated plant health monitoring.

General conclusion

The conclusion represents the final and most significant part of this research, drawing together the major theoretical insights and practical achievements established throughout the study. This work focused on the application of advanced deep learning techniques—particularly Convolutional Neural Networks (CNNs)—to the task of automated citrus disease detection. From the initial literature review to the development and deployment of the model, each phase of the project contributed to building a robust and intelligent system capable of accurately identifying multiple citrus leaf conditions.

The implemented model demonstrated excellent performance in terms of accuracy, precision, recall, and F1-score across all classes, thereby validating the effectiveness of the chosen architecture and training strategy. The successful classification of real-world test samples further reinforced the model’s generalization capabilities and its potential utility in practical agricultural settings.

Moreover, this research emphasized the importance of integrating technology with agriculture, presenting a scalable and data-driven solution that could assist farmers and agricultural experts in early disease detection and crop health monitoring. Such an approach can significantly reduce economic losses, minimize unnecessary pesticide use, and improve yield quality by enabling timely interventions.

Looking forward, several avenues exist for future work. These include expanding the dataset with more diverse samples under varying environmental conditions, incorporating additional citrus diseases, and exploring other deep learning architectures such as EfficientNet or Vision Transformers (ViTs). Further, the system could be enhanced with real-time mobile or drone-based deployment for in-field diagnostics. Integrating Internet of Things (IoT) devices and cloud computing could also contribute to building a fully automated, intelligent crop monitoring platform.

In summary, this research demonstrates the potential of combining artificial intelligence with agricultural science, offering a powerful tool for modern precision farming. It lays a solid foundation for future innovations aimed at achieving sustainable and efficient agricultural practices through the adoption of smart, technology-driven solutions.

References

- [1] Barbedo, J. G. A. (2018). Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification. *Computers and Electronics in Agriculture*, 153, 46-53 .
- [2] Wetterich, C. B., Neves, L. G., Belasque, J., & Marcassa, L. G. (2021). Detection of citrus huanglongbing using fluorescence imaging and deep learning. *Precision Agriculture*, 22(3), 711-727
- [3] Deng, X., Zhu, H., Chen, Y., & Lan, Y. (2022). HLB detection in citrus leaves using UAV-based multispectral imaging and a modified ResNet. *Remote Sensing*, 14(5), 1234.
- [4] Pires, R. D. L., Gonçalves, D. N., Oruê, J. P. M., et al. (2021). Black spot and canker classification in citrus plants using EfficientNet. *Sensors*, 21(18), 6339.
- [5] Boukralfa, F. Z, (2023). « Detection of plant diseases using deep learning based on image recognition », Master's thesis., university Abdelhamid Ibn Badis, Mostaganem .
- [6] Weizheng, S., Yachun, W., Zhanliang, C., & Hongda, W. (2008). « Grading method of leaf spot disease based on image processing ». International conference on computer science and software engineering (Vol. 6, pp. 491-494). IEEE.
- [7] Esma Aimeur, (2020) "Introduction to Artificial Intelligence," Department of Computer Science and Operations Research, University of Montreal (Canada).
- [8] Chapter 1, "Introduction to Artificial Intelligence," pages 4–5.
- [9] www.python.org/doc/essays/blurb/
- [10] The Jupyter Notebook — Jupyter Notebook 7.0.0b3 documentation (jupyter-notebook.readthedocs.io)
- [11] Deep Learning with Python SECOND EDITION FRANÇOIS CHOLLET
- [12] Introduction to Machine Learning with Python Andreas C. Müller, Sarah Guido
- [13] Python for Data Analysis, 2nd Edition Wes McKinney.
- [14] Zhang, C., & Kovacs, J. M. (2012). *The application of small unmanned aerial systems for precision agriculture: A review*. Precision Agriculture, 13(6), 693–712.

References

- [15] Hunt, E. R., Cavigelli, M., Daughtry, C. S., McMurtrey, J. E., & Walthall, C. L. (2010). *Evaluation of digital photography from model aircraft for remote sensing of crop biomass and nitrogen status*. Precision Agriculture, 6(4), 359–378.
- [16] Sankaran, S., Khot, L. R., & Esser, A. (2015). *UAV-based remote sensing for precision agriculture: A comprehensive review*. Computers and Electronics in Agriculture, 118, 66–73.
- [17] Zhang, T., Yang, J., & Yang, X. (2016). *Development and application of a variable-rate UAV spraying system*. Transactions of the Chinese Society of Agricultural Engineering, 32(13), 47–53.
- [18] Gago, J., Douthe, C., Coopman, R. E., Gallego, P. P., Ribas-Carbó, M., Flexas, J., & Escalona, J. M. (2015). *UAVs challenge to assess water stress for sustainable agriculture*. Agricultural Water Management, 153, 9–19.
- [19] Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). *Deep learning in agriculture: A survey*. Computers and Electronics in Agriculture, 147, 70–90.
- [20] Anderson, K., & Gaston, K. J. (2013). *Lightweight unmanned aerial vehicles will revolutionize spatial ecology*. Frontiers in Ecology and the Environment, 11(3), 138–146.
- [21] Tsouros, D. C., Bibi, S., & Sarigiannidis, P. G. (2019). *A review on UAV-based applications for precision agriculture*. Information, 10(11), 349.
- [22] Zarco-Tejada, P. J., Hubbard, N., & Loudjani, P. (2014). *Precision agriculture: An opportunity for EU farmers—potential support with the CAP 2014–2020*. European Parliament, Directorate-General for Internal Policies.
- [23] Honkavaara, E., Kaivosoja, J., Mäkynen, J., Pellikka, I., Pesonen, L., Saari, H., & Pölönen, I. (2013). *Hyperspectral reflectance signatures and point clouds for precision agriculture by light weight UAV imaging system*. ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences, II-1/W2.
- [24] Dutta, R., Sarmah, A., & Yadav, D. (2022). *Big data challenges in precision agriculture: UAV and satellite image processing*. Journal of King Saud University – Computer and Information Sciences.
- [25] Federal Aviation Administration (FAA). (2021). *Unmanned Aircraft Systems (UAS) Regulations & Policies*.
- [26] Haque, M. M., Islam, M. T., & Hassan, M. M. (2021). *Privacy and security issues in drone technology: A critical review*. Computers & Security, 103, 102130.
- [27] ang, L., & Shao, G. (2015). *Drone remote sensing for forestry research and practices*. Journal of Forestry Research, 26(4), 791–797.