People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

Larbi Tebessi University of Tebessa

Faculty of Exact Sciences and Natural and Life Sciences

Department of Computer Science

Final studies Thesis

For obtaining the Master's degree

Specialty: Information systems

Theme

Detection of plant diseases using artificial intelligence techniques

Realized by:

SALMI SOURIA

In front of the jury:

Pr. LAOUER MOUHAMED RIDDA	*PROF.*	*Larbi Tebessi University of Tebessa*	*President*
Dr. TABET KHAOULA	*MCB*	*Larbi Tebessi University of Tebessa*	*Examiner*
Dr. GHRIEB NAWEL	*MCB*	*Larbi Tebessi University of Tebessa*	*Supervisor*

University Year: 2024/2025

Acknowledgment

I would like to express my deepest gratitude first and foremost to my supervisor,

***Dr. GHRIEB NAWEL**, for her valuable advice and support.*

I also extend my thanks to all those whose presence, support, availability, and guidance gave me the courage to complete this project.

I would also like to express my sincere appreciation to the members of the jury, for agreeing to review and evaluate my work.

Special respect and gratitude to all the professors in the Computer Science Department for their invaluable contributions to this work.

Thank you all.

Dedication

To my beloved father Salmi L. and my dear mother Charef O,

Your unwavering love, prayers, and sacrifices have been the foundation of every success I have achieved. Your strength, patience, and boundless support have carried me through this journey. I dedicate this work to you with all the love and gratitude in my heart. May God reward you both and protect you always.

To my precious siblings Rabiaa, Fouzia, Domi, Khokha, Chawki, and Hamid, Faysal & Lotfi, and Hafa

Each of you has played a special role in my life, offering support, kindness, and encouragement that lifted me when I needed it most. I am truly blessed to have you by my side. This achievement is as much yours as it is mine.

To my dear friends,

Your support, motivation, and comforting words made even the toughest moments bearable. Thank you for believing in me.

To all my cherished family members,

Your love has been a source of strength and inspiration. I am forever grateful for each of you.

Abstarct

Artificial intelligence (AI), and in particular deep learning, is transforming the agricultural sector by enabling automated, fast, and reliable detection of plant diseases. This study explores an image classification approach for leaf images using Convolutional Neural Networks (CNNs), with the aim of automatically diagnosing several plant pathologies. The study is initially based on the standardized PlantVillage dataset, which is used to train and compare different architectures, including both pre-trained models (such as MobileNetV2) and a custom CNN model. The best model achieved an accuracy of 98% while maintaining low computational complexity.

To make this solution accessible to users, a web interface was also developed, allowing automated diagnosis from user-uploaded images. This work thus contributes to the design of AI-assisted diagnostic systems for more effective plant disease management.

Additionally, an initial series of tests was conducted on images from real-world conditions, marking an exploratory step toward validating the model in uncontrolled environments.

Keywords

Artificial Intelligence, Deep Learning, Convolutional Neural Network (CNN), Plant Disease Diagnosis, Smart Agriculture, Image Classification, Transfer Learning.

Résumé

L'intelligence artificielle (IA), et en particulier l'apprentissage profond, transforme le secteur agricole en permettant une détection automatisée, rapide et fiable des maladies des plantes. Cette étude explore une approche de classification d'images de feuilles à l'aide de réseaux de neurones convolutifs (CNN), dans le but de diagnostiquer automatiquement plusieurs pathologies végétales. L'étude repose initialement sur le jeu de données standardisé PlantVillage, utilisé pour entraîner et comparer différentes architectures, incluant à la fois des modèles pré-entraînés (tel que MobileNetV2) et un modèle CNN personnalisé. Le meilleur modèle a atteint une précision de 98 % tout en maintenant une faible complexité computationnelle.

Afin de rendre cette solution accessible aux utilisateurs, une interface web a également été développée, permettant le diagnostic automatisé à partir d'images chargées par l'utilisateur. Ce travail constitue ainsi une contribution à la conception de systèmes de diagnostic assistés par l'IA, pour une gestion plus efficace des maladies végétales.

En complément, une première série de tests a été réalisée sur des images provenant de conditions réelles, constituant une étape exploratoire vers la validation du modèle en environnement non contrôlé.

Mots-clés

Intelligence Artificielle, Apprentissage Profond, Réseau de Neurones Convolutif (CNN), Diagnostic des Maladies des Plantes, Agriculture Intelligente, Classification d'Images, Apprentissage par Transfert.

الملخص

يُحدث الذكاء الاصطناعي، وخاصةً التعلم العميق، نقلة نوعية في القطاع الزراعي من خلال تمكين الكشف الآلي والسريع والموثوق عن أمراض النبات. تستكشف هذه الدراسة نهجًا لتصنيف صور الأوراق باستخدام الشبكات العصبية التلافيفية (CNNs) لتشخيص العديد من أمراض النبات تلقائيًا. تعتمد الدراسة في البداية على مجموعة بيانات PlantVillage القياسية، المستخدمة لتدريب ومقارنة مختلف البنى، بما في ذلك النماذج المدربة مسبقًا (مثل MobileNetV2) ونموذج CNN مُخصص. حقق أفضل نموذج دقة بنسبة 98% مع الحفاظ على تعقيد حسابي منخفض.

لجعل هذا الحل متاحًا للمستخدمين، طُوّرت واجهة ويب تُمكّن من التشخيص الآلي من الصور التي يُحمّلها المستخدم. يُساهم هذا العمل بالتالي في تصميم أنظمة تشخيصية مدعومة بالذكاء الاصطناعي لإدارة أكثر كفاءة لأمراض النبات.

بالإضافة إلى ذلك، أُجريت سلسلة أولى من الاختبارات على صور من ظروف حقيقية، مما يُمثل خطوة استكشافية نحو التحقق من صحة النموذج في بيئة غير خاضعة للرقابة.

الكلمات المفتاحية

الذكاء الاصطناعي، التعلم العميق، الشبكات العصبية الالتفافية(CNN) ، كشف أمراض النباتات، الزراعة الذكية، تصنيف الصور، التعلم بالنقل.

Content Table

GENERAL INTRODUCTION **1**

Chapter 01: Plant Diseases **3**

1.1 Introduction 5

1.2 Agriculture 5

1.3 Plant diseases 7

1.4 Classes of plant diseases 8

1.4.1 Fungal Diseases 8

1.4.2 Bacterial plant diseases 10

1.4.3 Viral Plant Diseases 11

1.5 Agricultural Disease Diagnostic Systems 13

1.6 Apple diseases 14

1.6.1 Apple scab disease 14

1.6.2 Cedar-Apple Rust disease 16

1.6.3 Apple Black rot Disease 17

1.7 Conclusion 18

Chapter 02: Artificial intelligence **19**

2.1 Introduction 21

2.2 Artificial intelligence 21

2.3 Approaches of AI 22

2.3.1 Machine learning 23

2.3.2 Deep learning 24

2.4 Image recognition 24

2.4.1 Image File Formats 24

2.4.2 Image recognition tasks 25

2.4.2.1 Image Classification 25

2.4.2.2 Object Detection 26

2.4.2.3 Image Segmentation 28

2.5 Convolutional Neural Networks 29

2.5.1 Layers of a Convolutional Neural Network (CNN) 29

2.5.2 Advantages of CNNs in Deep Learning 31

2.5.3 Limitations of CNNs 32

2.5.4 Applications of Convolutional Neural Networks (CNNs) 32

2.6 Transfer Learning 33

2.6.1 Importance of Transfer Learning 33
2.6.2 Advantages of Transfer Learning 34
2.6.3 Common pre-trained models 34
2.7 Related works 37
2.8 Conclusion 41
Chapter 03: Experimental Protocol and Results 42
3.1 Introduction 43
3.2 Hardware and Software Environments 43
3.2.1 Tools and frameworks 43
3.2.2 Hardware Configuration 46
3.3 Methodology 47
3.4 Data preparation 48
3.4.1 Data Collection 48
3.4.2 Data preprocessing and augmentation strategy 50
3.5 Model Design and Training 51
3.5.1 Pretrained Models 52
3.5.2 Proposed Model (Custom CNN) 53
3.5.3 Models Training 54
3.6 Evaluation 55
3.6.1 Performance metrics 55
3.6.2 Obtained results 57
3.7 Exploratory evaluation of the model on real-world images 62
3.7.1 Experimental Protocol 63
3.7.2 Qualitative Analysis of Predictions Using Grad-CAM 66
3.8 Application Development 69
3.8.1 Tools and Frameworks Used 69
3.8.2 Backend of the application 70
3.8.3 Interface Design and Testing 72
3.9 Conclusion 74
General Conclusion and Future Work 76
Bibliography 78

Figures List

Figure 1.1: The Consequences of Plant Diseases on Agriculture and Society 7
Figure 1.2: Powdery Mildew..... 9
Figure 1.3: Downy Mildew 9
Figure 1.4: Bacterial Leaf Spot 10
Figure 1.5: Bacterial Wilt..... 11
Figure 1.6: Tomato Yellow Leaf Curl Virus..... 12
Figure 1.7: Mosaic Virus..... 12
Figure 1.8: Overview Diagram of a Plant Disease Diagnostic System..... 14
Figure 1.9: Apple leaf with early apple scab infections..... 15
Figure 1.10: Severe scab on apple..... 15
Figure 1.11: Cedar-apple rust on apple leaves 16
Figure 1.12: Black rot on leaves 17

Figure 2.1: Hierarchical Relationship Between AI, ML, and DL 22
Figure 2.2: Object Detection Example..... 27
Figure 2.3: Image segmentation example 28
Figure 2.4: Structure of CNN..... 30
Figure 2.5: ResNet architecture..... 34
Figure 2.6: VGG16 architecture 35
Figure 2.7: EfficientNet architecture..... 36
Figure 2.8: DenseNet architecture..... 36
Figure 2.9: MobileNET architecture 37

Figure 3.1: Python logo..... 44
Figure 3.2: Pytorch logo..... 44
Figure 3.3: Numpy logo. 44
Figure 3.4: Matplotlib logo 45
Figure 3.5: Scikit-learn logo. 45
Figure 3.6: Kaggle logo..... 46
Figure 3.7: Workflow of the Plant Disease Detection Methodology..... 47
Figure 3.8: Sample Leaf Images Extracted from the PlantVillage Dataset. 49
Figure 3.9: Samples of augmented images. 51

Figure 3.10: Custom CNN Architecture Diagram. 54
Figure 3.11: Representation of Confusion-matrix 56
Figure 3.12: Classification report of the CustomCNN. 59
Figure 3.13: Confusion Matrix (CustomCNN). 60
Figure 3.14: Examples of correctly labeled apple leaves. 61
Figure 3.15: Example of misclassified image. 61
Figure 3.16: Case 1 – Healthy Leaf (Original Image and Grad-CAM Map). 64
Figure 3.17: Case 2 – Apple Scab (Visible Lesion: Original Image and Grad-CAM) 64
Figure 3.18: Case 3 – Healthy Image Predicted as Diseased (Original Image and Grad-CAM) 65
Figure 3.19: Case 4 – Apple Scab (Original Image and Grad-CAM) 65
Figure 3.20: Case 5 – Black Rot (Original Image and Grad-CAM) 66
Figure 3.21: Homepage of our web interface. 72
Figure 3.22: Examples Result for Different Apple Leaf Conditions. 73

Tables List

Table 1.1: Comparative table between traditional and modern farming 6

Table 2.1: Definitions of Artificial Intelligence (AI) 22
Table 2.2: Categories of Machine Learning and Their Algorithms 23
Table 2.3: Common Image File Formats and Their Descriptions 25
Table 2.4: Comparative Table of Recent Research 39

Table 3.1: Classes of apple diseases. 49
Table 3.2: Comparative Characteristics of Selected CNN Architectures. 52
Table 3.3: Hyperparameter search space and final selected configuration. 54
Table 3.4: Summarized comparison of all models. 57
Table 3.5: Summary of Model Predictions and Grad-CAM Interpretations. 67

Abbreviations List

AI	Artificial Intelligence
TR4	Tropical Race 4
TYLCV	Tomato Yellow Leaf Curl Virus
CNNs	Convolutional Neural Networks
FRAC	Fungicide Resistance Action Committee
ML	Machine Learning
DL	Deep Learning
k-NN	k-Nearest Neighbors
SVM	Support Vector Machines
PCA	Principal Component Analysis
DQN	Deep Q-Networks
RNNs	Recurrent Neural Networks
DBNs	Deep Belief Networks
JPEG	Joint Photographic Experts Group
GIF	Graphics Interchange Format
PNG	Portable Network Graphics
TIFF	Tagged Image File Format
BMP	Bitmap
PDF	Portable Document Format
ReLU	Rectified Linear Unit
TPU	Tensor Processing Unit
GPU	Graphics Processing Unit
X-ray	X-ray (used in medical diagnostics)
MRI	Magnetic Resonance Imaging
VGG	Visual Geometry Group
ResNet	Residual Network
DCNN	Deep Convolutional Neural Network
B0, B3, B7	Variants of EfficientNet models
DCGAN	Deep Convolutional Generative Adversarial Network
NST	Neural Style Transfer
BIM	Basic Iterative Method

CLAHE	Contrast Limited Adaptive Histogram Equalization
AMF	Adaptive Median Filtering
FC	Fully Connected
MB	Mega Byte
TP	True Positives
FP	False Positives
FN	False Negatives
UI	User Interface
PIL	Python Imaging Library
CSS	Cascading Style Sheets
VS Code	Visual Studio Code

GENERAL INTRODUCTION

GENERAL INTRODUCTION

Agriculture remains a fundamental pillar of global development, supplying essential food resources and sustaining the livelihoods of billions worldwide. Despite its critical importance, the agricultural sector continues to face significant challenges, particularly plant diseases, which substantially reduce both the quantity and quality of crop yields. These diseases not only cause severe economic losses but also pose a serious threat to global food security, especially in regions where agriculture is the primary source of income and employment.

The early and accurate diagnosis of plant diseases is crucial for effective crop management, timely treatment, and the implementation of sustainable agricultural practices. However, traditional diagnostic approaches, which depend heavily on manual inspection by trained experts, are often time-consuming, subjective, and impractical for large-scale or real-time monitoring.

In response to these limitations, recent advances in Artificial Intelligence (AI) particularly Deep Learning have demonstrated significant potential in automating the classification of plant diseases. Among these, Convolutional Neural Networks (CNNs) have emerged as powerful tools for image-based classification tasks. These models are capable of learning intricate visual patterns directly from leaf images, thus enabling the accurate identification of various plant diseases.

This study explores the application of deep learning techniques for the automated diagnosis of diseases in apple leaves, using image classification models based on Convolutional Neural Networks (CNNs). Multiple architectures have been employed, including well-established pre-trained models such as MobileNetV2, EfficientNetB0, and ResNet18, alongside a custom-designed CNN architecture tailored to the task. The primary aim is not only to enhance diagnostic accuracy, but also to develop a lightweight yet precise model that can be effectively deployed in resource-constrained environments, such as mobile devices.

This thesis is organized into three chapters.

The first chapter provides essential background on agriculture and plant pathology, highlighting the impact of plant diseases on crop productivity and emphasizing the importance of early diagnostic. It also introduces apple diseases.

The second chapter presents the theoretical foundation of the study, offering an overview of Artificial Intelligence, Machine Learning, and Deep Learning, with a focus on Convolutional Neural Networks, also transfer learning. A review of recent state-of-the-art research is also included to position this work within the current scientific landscape.

The third chapter presents the experimental framework and implementation details of the apple leaf disease classification system developed in this study. It explains the data preparation process using the PlantVillage dataset, the methodology adopted, and the hardware and software environments. The chapter explores the development and training of four deep learning models: ResNet18, MobileNetV2, and EfficientNetB0 and a custom Convolutional Neural Network (CNN) built from scratch. Evaluation metrics are applied to assess the models performance, and Grad-CAM is used on real world images to visualize model decisions and enhance interpretability. The chapter concludes with a discussion on the effectiveness of the models and introduces a web based application that enables real time diagnosis of apple leaf diseases using the best performing model.

Chapter 01

Plant Diseases

2.1 Introduction

Agricultural production plays a vital role in economic development, yet crops are highly vulnerable to infections caused by various pathogens. These infections can significantly reduce yields or even lead to plant death, posing a serious threat to food security and economic stability. Effective diagnosis of plant diseases is essential to mitigating these risks, and artificial intelligence has emerged as a powerful tool in this field. Implementing disease management strategies, including routine crop monitoring and precise diagnosis is crucial for minimizing the adverse effects of plant diseases on agricultural productivity.

This chapter explores the significance of agriculture, comparing traditional and modern farming. It also delves into plant diseases, their symptoms, underlying causes. Furthermore, special attention is given to agricultural disease diagnostic systems, and to apple diseases, as apples are among the most widely cultivated and economically important crops.

2.2 Agriculture

Over the past several decades, agriculture has experienced remarkable advancements, with global agricultural output expanding nearly fourfold between 1961 and 2020. This substantial growth has been fueled by increased investments in agricultural research and technological innovation, both from public institutions and private enterprises. These investments have led to the development of more efficient farming techniques, improved crop varieties, and enhanced irrigation systems, all of which have significantly boosted productivity. As a result, agricultural expansion has played a crucial role in addressing food security challenges, supporting rural economies, and reducing poverty, particularly in developing regions where a large portion of the population depends on farming for their livelihood [1].

With the acceleration of globalization and the emergence of modern technologies, traditional farming practices have become increasingly inadequate in addressing the growing complexities of food production. In response to these challenges, many farmers have turned to precision agriculture, a modern approach that leverages digital tools and data analytics to enhance productivity, reduce waste, and optimize resource use. This transition has significantly reshaped agricultural systems by shifting the focus from small-scale, labor-intensive farming toward large-scale agribusinesses capable of integrating innovative technologies on a broader scale.

Moreover, continuous technological progress has fueled this transformation, propelling the agricultural sector from conventional methods rooted in manual labor and experience to smart farming a data-driven, automated, and efficiency-oriented model. The table 1.1 below highlights the key differences between traditional and modern farming systems [2].

Table 1.1: Comparative table between traditional and modern farming.

Feature	Traditional Farming	Modern Farming
Approach	Follows age-old techniques using basic tools.	Integrates advanced science and technology.
Equipment & Methods	Utilizes manual labor, animal power, and natural methods.	Implements mechanized tools, automated systems, and AI-driven farming.
Soil & Crop Management	Uses organic compost and traditional pest control.	Applies synthetic fertilizers and chemical treatments for higher productivity.
Scale of Production	Primarily small-scale for self-sustenance.	Large-scale farming aimed at commercial markets.
Productivity & Efficiency	Lower yields, labor-intensive, and time-consuming.	High efficiency with increased crop yields.
Environmental Considerations	Minimal environmental harm but limited output.	Greater efficiency but potential ecological risks.
Climate Dependence	Heavily reliant on natural conditions.	Uses modern irrigation, controlled environments, and climate monitoring.
Purpose	Primarily for personal or local consumption.	Designed to meet global food demands and ensure food security.

However, this evolution in the agricultural world does not protect crops from the many biological threats, particularly plant diseases, which remain a major source of agricultural losses.

2.3 Plant diseases

Plants are highly diverse organisms that vary significantly in size, structure, and complexity, ranging from microscopic algae to towering trees. They play a fundamental role in sustaining life on Earth through photosynthesis, a process that converts sunlight into energy while releasing oxygen into the atmosphere. In addition to serving as the primary source of food for humans and animals, plants provide essential raw materials such as fibers, medicinal compounds, construction materials, and valuable natural substances like oils and latex. Beyond their economic and ecological importance, plants contribute to environmental balance by improving air quality, preventing soil erosion, and supporting biodiversity [3].

Plant diseases have profoundly affected global agriculture, leading to food crises, economic instability, and major social disruptions. Outbreaks of diseases such as rusts, blights, spots, and mosaics have devastated crops, causing consequences similar to natural disasters. Throughout history, these outbreaks have contributed to famines, malnutrition, and large-scale migrations (Figure 1.1).

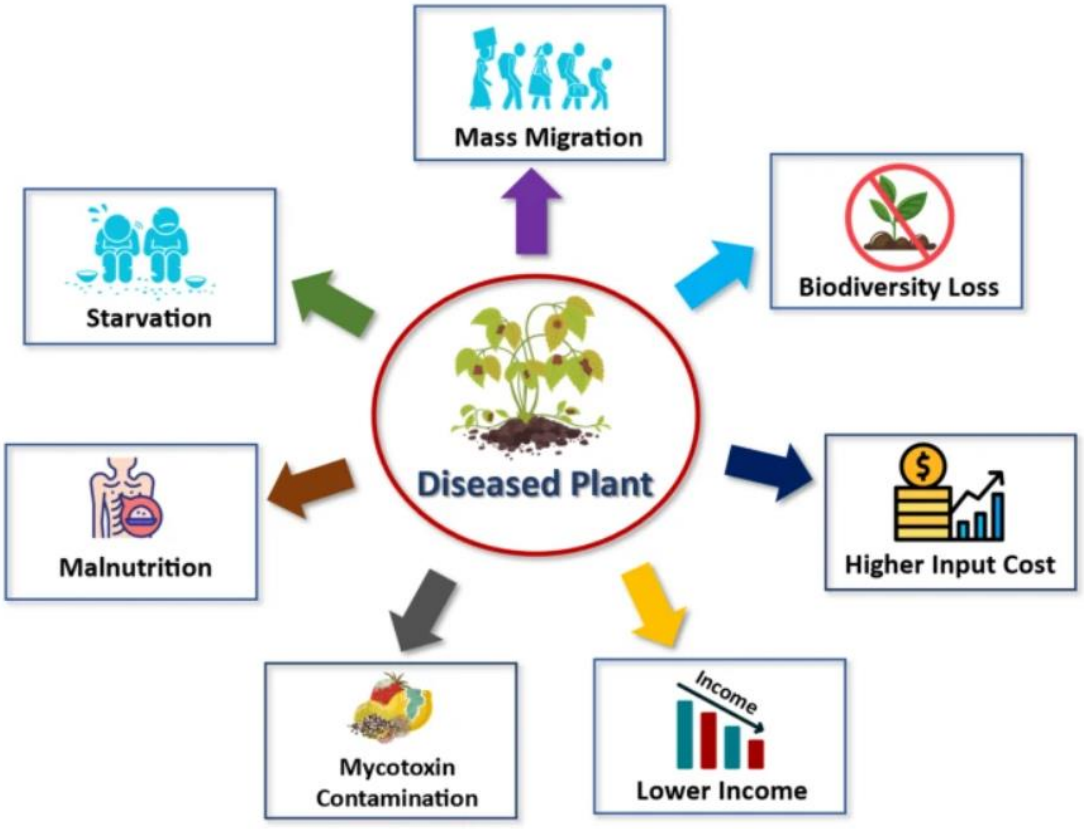


Figure 1.1: The Consequences of Plant Diseases on Agriculture and Society [3].

A significant example is the Irish Potato Famine in the 19th century, where potato blight (*Phytophthora infestans*) caused widespread crop failure, leading to the deaths of around 1.1 million people and forcing 2 million more to emigrate. Similarly, the Bengal Famine of 1943, triggered by brown spot disease in rice, resulted in the deaths of 3 million people. In the 1990s, the East African cassava mosaic virus led to severe crop losses, worsening food insecurity in already struggling communities.

Beyond agriculture, plant diseases have also caused significant ecological damage. The chestnut blight nearly wiped out mature American chestnut trees, dramatically altering North American forests. Dutch elm disease led to the decline of elm trees across multiple continents. Additionally, Panama wilt devastated the Gros Michel banana variety, and the ongoing threat of Tropical Race 4 (TR4) endangers the widely cultivated Cavendish banana.

These cases highlight the devastating impact of plant diseases on food production, ecosystems, and human societies, underscoring the need for effective disease control measures and sustainable agricultural practices [4].

2.4 Classes of plant diseases

Plant diseases can be classified into several categories based on the nature of the pathogens responsible for the infection. The most common classes include fungal, bacterial, and viral diseases, each affecting plants in unique ways.

2.4.1 Fungal Diseases

Fungal Diseases are infections in plants caused by various types of fungi. These diseases often appear as molds on leaves, stems, or fruits.

Powdery Mildew

Powdery mildew often appears as a white, dust-like coating on the surface of leaves (figure 1.2), young stems, and shoots. It usually starts on the upper side of the leaves and gradually spreads. Infected areas may begin to yellow, and over time, without treatment, the leaves can turn brown and die [5].



Figure 1.2: Powdery Mildew [5]

Downy Mildew

Downy mildew typically shows up as pale yellow or whitish spots on the upper surfaces of leaves and sometimes on stems (figure 1.3). On the underside of the leaves, a soft gray or fuzzy mold may develop. This disease can also interfere with a plant's ability to bloom. As it progresses, affected leaves may dry out, turn brown, and eventually fall off [5].



Figure 1.3: Downy Mildew [5]

2.4.2 Bacterial plant diseases

Bacterial Plant Diseases are caused by harmful bacteria that invade plant tissues, often through wounds or natural openings. These infections can lead to symptoms such as wilting, leaf spots, soft rots.

Bacterial Leaf Spot

Bacterial leaf spot is a disease that leads to the appearance of discolored spots typically yellow, brown, or black on a plant's leaves (figure 1.4).These spots often look water-soaked and tend to spread across the foliage. In the early stages, the symptoms usually begin on the leaves and fruit, starting off small before potentially expanding [6].



Figure 1.4: Bacterial Leaf Spot [6].

Bacterial Wilt

This disease originates from the soil and is mainly caused by the bacterium Ralstonia solanacearum. It affects a wide variety of host plants, particularly key crops such as tomatoes, potatoes, eggplants, and peppers. The infection targets the plant's vascular tissues (figure 1.5), blocking the flow of water and nutrients, which results in significant wilting and can eventually cause the plant to die [7].



Figure 1.5: Bacterial Wilt [7]

2.4.3 Viral Plant Diseases

Viral Plant Diseases are caused by plant viruses that infect cells and interfere with normal growth and function. These diseases often result in symptoms like yellowing, mosaic patterns on leaves, stunted growth, or deformed fruits. Viruses are usually spread by insects such as whiteflies.

Tomato Yellow Leaf Curl Virus

Tomato Yellow Leaf Curl Virus (TYLCV) affects more than 30 plant species, but it is particularly notorious for severely impacting tomato crops, sometimes leading to complete yield loss. Both greenhouse and open-field tomato plants are vulnerable to this virus. TYLCV is transmitted by the silverleaf whitefly (*Bemisia tabaci*) [8]. Infected tomato plants often show noticeable symptoms such as reduced leaf size, yellowing between the veins, and leaves curling upward and inward toward the center (figure 1.6).



Figure 1.6: Tomato Yellow Leaf Curl Virus [8]

Mosaic Virus

There are multiple types of mosaic viruses, with the most common being tomato mosaic virus and tobacco mosaic virus. It infects a variety of plants, including tomatoes, peppers, potatoes, apples, pears, cherries, cucumbers, lettuce, and petunias. Infected plants display mottled yellow and green leaves, leaf curling, stunted growth, malformed fruits, and reduced yield, particularly in hot weather (figure 1.7) [9].



Figure 1.7: Mosaic Virus [9]

2.5 Agricultural Disease Diagnostic Systems

Timely and precise diagnostic of plant diseases is essential for ensuring sustainable agricultural productivity and preventing significant crop losses. Disease outbreaks present one of the most serious challenges in modern farming, often leading to substantial economic damage. Early identification and intervention play a crucial role in mitigating these risks, highlighting the need for efficient and reliable disease detection methods [10].

An agricultural disease prediction system is a management tool used to forecast the emergence or worsening of diseases affecting cultivated plants [11]. Farmers rely on these systems to make informed and cost-effective decisions regarding disease control measures. Typically, such systems guide farmers through a series of questions about the susceptibility of the host crop and provide recommendations based on both current and anticipated weather conditions. These recommendations generally aim to evaluate whether treatment for the identified disease is necessary.

In this context, the primary role of a plant disease diagnostic system is to accurately identify the early signs of disease onset, enabling timely interventions. This process usually involves multiple stages, as illustrated in (figure 1.8), allowing growers to make effective decisions regarding the use of plant protection products [11]:

1. **Image Acquisition:** Capturing high quality images of plant leaves using devices like smartphones or digital cameras. The quality and variety of these images significantly influence the diagnostic system's performance.
2. **Preprocessing:** Enhancing the captured images to reduce noise, normalize lighting conditions, and resize to standard dimensions. This step ensures consistency and improves the model's ability to learn from the data.
3. **Feature Extraction:** Identifying and isolating key patterns, textures, or color distributions indicative of specific plant diseases. In deep learning systems, this is often performed automatically by convolutional layers in a CNN, allowing the model to learn hierarchical features relevant to disease symptoms.
4. **Disease Classification:** Analyzing the extracted features to predict the health status of the plant. If diseased, the system identifies the specific type of infection, enabling timely and appropriate intervention measures.

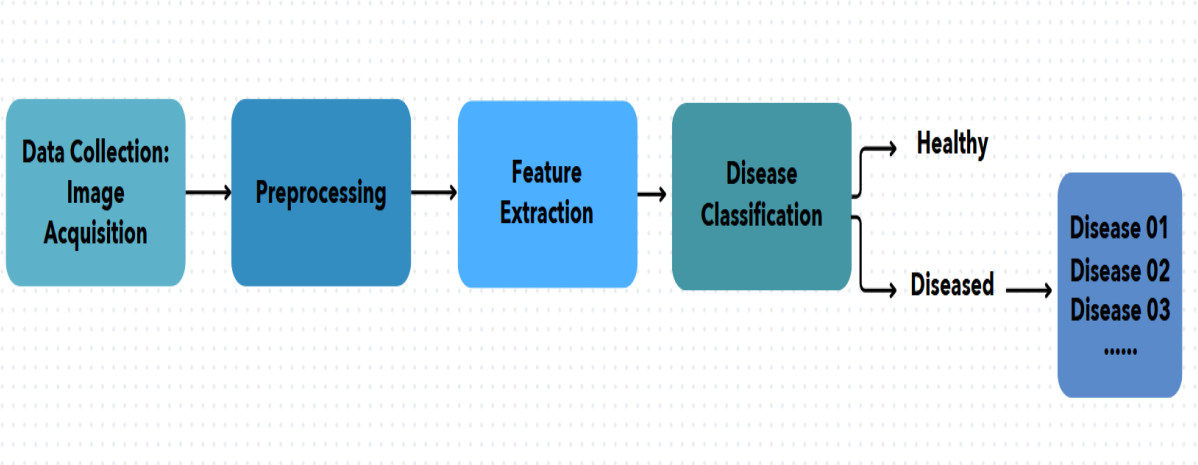


Figure 1.8: Overview Diagram of a Plant Disease Diagnostic System.

2.6 Apple diseases

Among the many cultivated species, the apple tree is one of the most susceptible to various fungal infections. Its economic importance justifies special attention in AI-assisted diagnostic systems.

In this study, we will focus on a specific fruit-bearing plant: the apple tree (*Malus domestica*). Apples are among the most cultivated and consumed fruits worldwide, valued for both their nutritional benefits and economic significance. Apple trees are generally planted during the dormant season, typically in late fall or early spring, and begin producing fruit after a few years. They grow best in temperate regions and require well-drained, fertile soil. Consistent irrigation and proper care during flowering and fruit development stages are essential to ensure healthy growth and high quality yields.

2.6.1 Apple scab disease

Apple scab is a widespread and damaging disease that impacts apple trees. It is caused by the fungus *Venturia inaequalis*, which mainly infects the leaves and fruits, potentially resulting in considerable reductions in both crop yield and quality if left unmanaged [13].

Apple scab symptoms usually begin to show on the foliage as small, velvety, olive-colored spots, particularly around the blooming stage (figure 1.9). As the infection advances, these spots grow larger, turn dark, and may lead to early leaf yellowing and drop. On the fruit, the disease presents as grayish to black patches that can expand, crack, and develop a rough, cork-like texture (figure 1.10). This can lead to misshapen apples and significantly lower their

commercial value. Infections that occur later in the season may result in tiny dark specks, often referred to as "pinpoint scab," which can appear during fruit storage.



Figure 1.9: Apple leaf with early apple scab infections [13]



Figure 1.10: Severe scab on apple.

Control Strategies

Effective control of apple scab requires an integrated approach. Sanitation practices, such as removing and destroying fallen leaves in autumn or early spring, are essential to eliminate sources of overwintering inoculum. Pruning the trees regularly improves air circulation, which facilitates faster drying of leaves and reduces the likelihood of infection [13].

2.6.2 Cedar-Apple Rust disease

Cedar-apple rust is a fungal infection that results from the relationship between apple trees, which are part of the Rosaceae family, and cedar trees, such as the Eastern red cedar [14].

After blooming, apple leaves begin to show bright orange-yellow lesions (figure 1.11). These lesions gradually grow larger and become more prominent during July and August. In some varieties of cedar apple rust, similar lesions can also appear on the fruit.



Figure 1.11: Cedar-apple rust on apple leaves [14]

Control Strategies

To protect the leaves and fruit of apple varieties susceptible to cedar-apple rust, fungicides are essential. These fungicides should be applied during prolonged wet weather, usually from the pink bud stage to the second cover stage [14].

2.6.3 Apple Black rot Disease

Black rot is a fungal infection triggered by *Botryosphaeria obtusa*, which commonly affects apple trees and other members of the pomaceous family. The disease impacts various parts of the tree including leaves, bark, and fruit resulting in signs like frogeye-patterned leaf spots, depressed cankers on branches, and rotting of the fruit [15].

The initial signs of black rot on leaves are small, purple spots (figure 1.12) that gradually expand into circular lesions featuring purple borders and tan to yellowish-brown centers, creating a characteristic “frogeye” pattern.

In some cases, tiny black fungal structures called pycnidia can be seen at the center of these spots. On branches and limbs, the disease causes reddish-brown sunken cankers that may increase in size over the years. These older cankers often have raised, lobed edges, and the bark in their center tends to become pale, flaky, and detached. As for the fruit, infection typically begins at the calyx end or enters through wounds, producing dark brown to black areas often arranged in concentric rings. Despite the decay, the affected flesh remains tough and leathery. Eventually, the infected fruit dries out, shrivels, and forms a mummy-like appearance, often covered in pycnidia [15].



Figure 1.12: Black rot on leaves [15]

Control Strategies

Effective control of black rot involves several cultural and chemical practices. Protecting apple trees from environmental stresses such as drought, winter injury, and insect damage is essential to reduce susceptibility. Infected twigs, branches, cankers, and mummified fruit should be removed and properly disposed of to minimize sources of fungal spores [15].

2.7 Conclusion

This chapter highlighted the fundamental role of agriculture in economic and societal development and key differences between traditional & modern farming, while emphasizing the vulnerability of crops to a wide range of plant diseases. These diseases, often caused by bacteria, fungi, or viruses, can drastically impact crop yields and threaten food security on both local and global scales. The chapter reviewed the symptoms and causes of common plant diseases, as well as the importance of early and accurate diagnosis to ensure effective treatment. Special attention was given to agricultural disease diagnostic systems and to diseases affecting apple trees.

Building on the critical challenges outlined in crop disease management, the next chapter delves into how Artificial Intelligence offers innovative solutions to enhance diagnosis.

Chapter 02
Artificial intelligence

4.1 Introduction

Artificial Intelligence (AI) is transforming various industries, including agriculture, by enabling machines to learn from data and make informed decisions. Key AI subfields like machine learning (ML) and deep learning (DL) have revolutionized image recognition and predictive analysis, playing a vital role in plant disease diagnosis.

This chapter provides a comprehensive overview of artificial intelligence, with a particular emphasis on machine learning, deep learning, convolutional neural networks (CNNs), and transfer learning. It also includes a review of related work, comparing recent advances in AI-based approaches to plant disease diagnosis.

4.2 Artificial intelligence

Artificial intelligence (AI) is a rapidly advancing field with the potential to transform various aspects of daily life. It is widely applied in areas such as self-driving cars, virtual assistants, medical diagnosis, and financial forecasting. Additionally, AI is revolutionizing agriculture by enhancing crop monitoring, optimizing irrigation, classifying plant diseases, and improving overall farm productivity through automation and predictive analytics [16].

The term **"artificial"** originates from the Latin *artificialis*, derived from *artificium*, meaning "skill" or "craftsmanship." It stems from *artifex*, a combination of *ars* ("art") and *facere* ("to make"), signifying something created through human ingenuity, often as a replica of what exists naturally [17].

The word **"intelligence"** comes from the Latin *intelligentia*, which is based on *intelligere*, meaning "to understand" or "to comprehend." This term is formed from *inter* ("between") and *legere* ("to choose" or "to read"), reflecting the ability to analyze and make decisions. Intelligence broadly refers to the capacity for reasoning, learning, problem-solving, and adapting to new circumstances [18].

When combined, **Artificial Intelligence (AI)** represents the development of systems or machines designed to replicate human cognitive abilities. AI enables machines to perform tasks such as recognizing patterns, making decisions, and solving problems functions traditionally associated with human intelligence (table 2.1).

Table 2.1: Definitions of Artificial Intelligence (AI).

Definition	Source
AI refers to the capability of machines to replicate complex human behaviors and skills.	[19]
AI is the ability of machines to execute cognitive functions similar to those of the human mind, including perception, reasoning, learning, interaction with the environment, and problem solving.	[20]
Artificial Intelligence enables systems to learn from previous experiences, adjust to new inputs, and carry out tasks that typically require human intelligence.	[21]
AI, also known as machine intelligence, is a form of intelligence exhibited by machines, distinguishing it from the natural intelligence observed in humans and animals.	[22]

4.3 Approaches of AI

Figure 2.1 illustrates the hierarchical relationship among Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL), where DL is considered a specialized subset of ML, which itself falls under the broader domain of AI.

The upcoming sections explore the key functionalities of artificial intelligence, including Machine Learning (ML) and Deep Learning (DL).

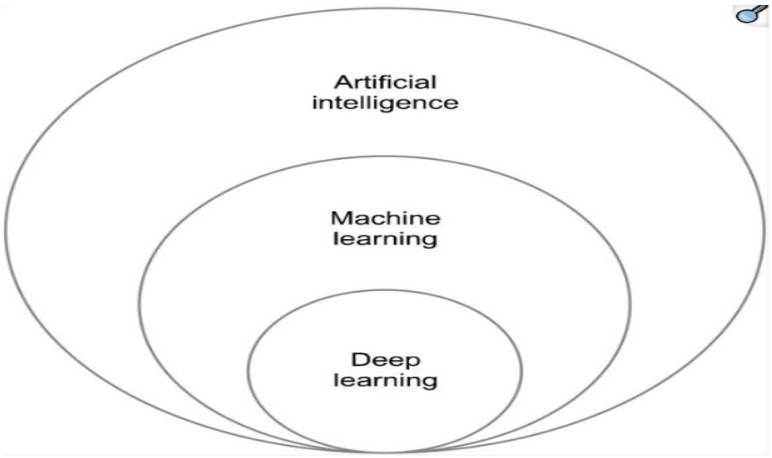


Figure 2.1: Hierarchical Relationship Between AI, ML, and DL [23].

4.3.1 Machine learning

Machine Learning (ML) is a branch of artificial intelligence that focuses on developing algorithms capable of learning from data to make predictions or decisions without being explicitly programmed (Table 2.2). These algorithms detect patterns within data, allowing systems to refine their performance and improve over time [24].

ML algorithms are broadly classified into four main categories:

1. **Supervised Learning**: Algorithms that learn from labeled datasets, enabling them to categorize data and make accurate predictions.
2. **Unsupervised Learning**: Algorithms designed to analyze unlabeled data, discovering hidden patterns and structures without predefined guidance
3. **Semi-Supervised Learning**: Algorithms that leverage a small amount of labeled data along with a larger set of unlabeled data to enhance learning accuracy.
4. **Reinforcement Learning**: Algorithms that learn optimal decision-making strategies by interacting with an environment, receiving rewards or penalties, and adjusting their actions accordingly.

Table 2.2: Categories of Machine Learning and Their Algorithms.

Category	Examples of Algorithms
Supervised Learning	k-Nearest Neighbors (k-NN), Support Vector Machines (SVM), Decision Trees, Linear Regression
Unsupervised Learning	k-Means Clustering, Hierarchical Clustering, Principal Component Analysis (PCA)
Semi-Supervised Learning	Self-Training, Co-Training, Graph-Based Methods
Reinforcement Learning	Q-Learning, Deep Q-Networks (DQN), Policy Gradient Methods

4.3.2 Deep learning

Deep Learning (DL) is an advanced subfield of machine learning that utilizes multi-layered artificial neural networks to analyze and interpret complex data patterns. These deep architectures enable the automatic extraction of meaningful features from raw input data, making them highly effective for classification, prediction, and pattern recognition. The growing success of DL is largely due to improvements in data collection, storage, and processing capabilities, allowing for more sophisticated learning and adaptation [23].

Several fundamental deep learning architectures and techniques include [23]:

- **Convolutional Neural Networks (CNN)**: Designed for structured data like images, CNNs use convolutional layers to detect spatial hierarchies of features, enabling tasks such as image classification and object detection.
- **Recurrent Neural Networks (RNN)**: Specialized for sequential data, RNNs incorporate memory through loops in their structure, making them suitable for tasks like language modeling, speech recognition, and time series analysis.
- **Deep Belief Networks (DBN)**: These consist of multiple layers of probabilistic, latent variables trained in a layer wise manner, often used for unsupervised learning and feature extraction.

4.4 Image recognition

Image recognition is a branch of computer vision and artificial intelligence that enables systems to analyze and identify objects, locations, individuals, text, and activities within images. It involves diagnosing and distinguishing specific elements in digital images or videos. This technology is widely applied in areas such as facial recognition, object detection, and image classification [25].

4.4.1 Image File Formats

Table 2.3 presents an overview of the most common image file formats used in digital applications. Each format has its own characteristics in terms of compression, quality, and suitability for specific tasks.

Table 2.3: Common Image File Formats and Their Descriptions [26].

Abbreviation	Format Name	Description
JPEG	Joint Photographic Experts Group	A widely used format that applies lossy compression to reduce file size while maintaining acceptable image quality. It is commonly used for photographs and web graphics but may degrade in quality with multiple edits.
GIF	Graphics Interchange Format	A format known for its support of simple animations and a limited color palette of up to 256 colors. It uses lossless compression and supports transparency, making it suitable for web-based graphics.
PNG	Portable Network Graphics	A lossless image format that preserves high-quality visuals and supports transparent backgrounds. It is ideal for digital graphics, logos, and images requiring sharp details without quality loss.
TIFF	Tagged Image File Format	A high quality format often used in professional photography and publishing. TIFF images use lossless compression, preserving all details, but they result in large file sizes and may not be universally supported across all applications.
BMP	Bitmap	An uncompressed raster format primarily used in Windows applications. Due to its lack of compression, BMP files tend to be large, making them less practical for web use but useful in certain design and editing environments.
PDF	Portable Document Format	A format developed by Adobe to store images, text, and graphics in a fixed layout. PDF files are widely used for document sharing, ensuring that content appears the same regardless of the device or software used to open them.

4.4.2 Image recognition tasks

Image recognition encompasses several key tasks that enable computers to interpret and process visual information effectively. These tasks include image classification, object detection, and image segmentation.

4.4.2.1 Image Classification

Image classification is a key task in computer vision that involves assigning a predefined category or label to an image or video. The primary objective is to accurately classify images into specific classes by training a machine learning model on a labeled dataset. Once the model has been trained, it can predict the category of new, unseen images [27].

There are two primary types of image classification [27]:

1. **Single-label classification**: This approach assigns an image to only one category, making it suitable for cases where classes are mutually exclusive. For example, an image of an animal could be classified as either a cat or a dog, but not both.
2. **Multi-label classification**: Unlike single-label classification, this method allows an image to be categorized under multiple labels simultaneously. This is beneficial when an image contains several objects or attributes. For instance, in interior design photography, a single image may include multiple elements such as beds, tables, chairs, and paintings, all of which need to be identified and labeled.

Image classification plays a crucial role in various domains, including [27]:

- **Digital asset management**: AI-powered models can automatically tag and organize media files, simplifying content management.
- **Content moderation**: AI is used to monitor and filter user-generated content, ensuring that explicit, illegal, or inappropriate images are detected and removed.
- **Product categorization**: In e-commerce, classification models help categorize products into appropriate groups, such as distinguishing between different types of lighting fixtures like pendant lights and sconces.

Advanced Uses of Image Classification

Image classification can be combined with other machine learning techniques to enhance its capabilities. A common approach is integrating it with object detection, where a model first detects an object within an image and then classifies it into relevant subcategories. This approach is widely used in areas like autonomous driving, medical diagnostics, and retail analytics [27].

4.4.2.2 Object Detection

Object detection is a computer vision technique that identifies and locates multiple objects within an image (figure 2.2) or video. Unlike simple classification, object detection provides precise spatial information by drawing bounding boxes around detected objects. This technique is particularly useful for identifying specific objects within a scene, such as vehicles on a road or individuals in a crowd. Typically, detection models first categorize objects into broad groups

before classifying each instance into more specific subcategories. For example, a model may initially detect a vehicle and then classify its make and model [27].

Applications of Object Detection

Object detection is widely used across various fields to analyze digital images and videos, distinguishing objects and examining their characteristics. Some key applications include [27]:

- **People and vehicle detection and tracking**: AI models can identify individuals and moving vehicles, assigning unique identifiers to track their movement across multiple video frames.
- **Demographic analysis**: Facial detection models can recognize faces and classify attributes such as estimated age, gender, and ethnicity.
- **Satellite image analysis**: Object detection is utilized to examine satellite images, identifying structures or changes over time, such as monitoring aircraft at airports, assessing land development, or analyzing economic activities in urban areas.

By enabling automated recognition and localization of objects, object detection plays a critical role in fields like security surveillance, traffic monitoring, healthcare, and environmental studies.

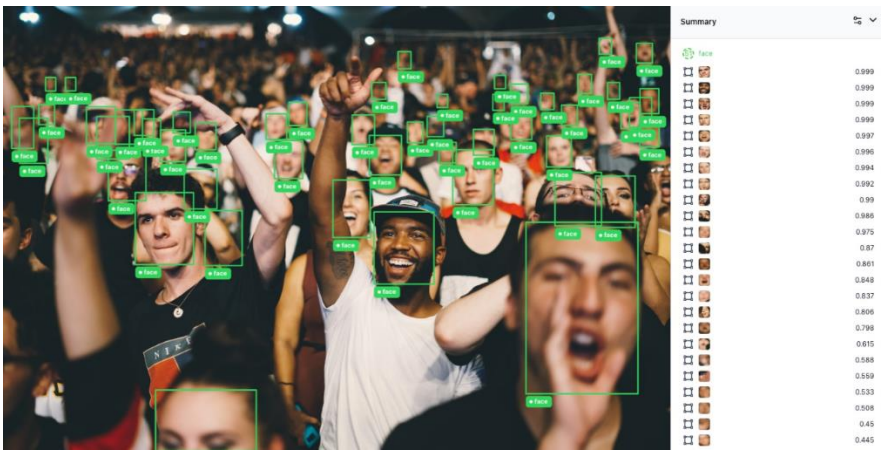


Figure 2.2: Object Detection Example [27]

4.4.2.3 Image Segmentation

Image segmentation is a technique in computer vision that assigns a label to each pixel in an image, allowing for detailed, pixel-level recognition of objects (figure 2.3). Unlike traditional object detection, which relies on bounding boxes, segmentation provides a more precise outline of objects by distinguishing individual pixels associated with specific elements in an image [27].

There are two main types of image segmentation [27]:

- **Semantic Segmentation**: This method groups all pixels belonging to the same object category together. For instance, in an image containing multiple vehicles, all pixels corresponding to any car might be assigned the same color. This approach is particularly useful in applications like autonomous driving, where identifying general object categories is more critical than distinguishing individual instances.



Figure 2.3: Image segmentation example [28].

- **Instance Segmentation**: Unlike semantic segmentation, instance segmentation differentiates between individual objects of the same category. For example, in an image with multiple cars, this technique labels each car separately rather than assigning them a single group label. This is beneficial in scenarios where distinguishing between different instances of an object is necessary, such as tracking multiple people or vehicles in a given scene.

Applications of Image Segmentation

Image segmentation is widely applied across various industries to enhance visual analysis and decision-making. Some of its key applications include [27]:

- **Robotic Vision**: Used in fields such as industrial automation and medical robotics, image segmentation enables robots to interpret their environment with greater precision, improving their ability to interact with objects.
- **Autonomous Driving**: Self-driving vehicles rely on segmentation to detect and differentiate between road elements, such as other vehicles, pedestrians, lane markings, and traffic signs.
- **Industrial Inspection**: In manufacturing, segmentation aids in detecting product defects, surface anomalies, or equipment damage, such as identifying rust, cracks, or missing components.
- **Medical Imaging**: In healthcare, segmentation techniques help medical professionals analyze medical scans more accurately. This includes identifying tumors, segmenting organs, mapping blood vessels, and detecting neurological disorders through brain imaging.

4.5 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a specialized deep learning model designed for processing visual data. It excels in handling images and videos, making it a fundamental tool for various computer vision applications. Due to its strong ability to identify objects, CNNs are widely employed in tasks such as image and object recognition. Common applications include autonomous vehicles, facial recognition systems, and medical image analysis [29].

4.5.1 Layers of a Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are composed of multiple specialized layers that work together to process and interpret image data (figure 2.4). Each layer plays a distinct role in feature extraction, dimensionality reduction, and classification [29].

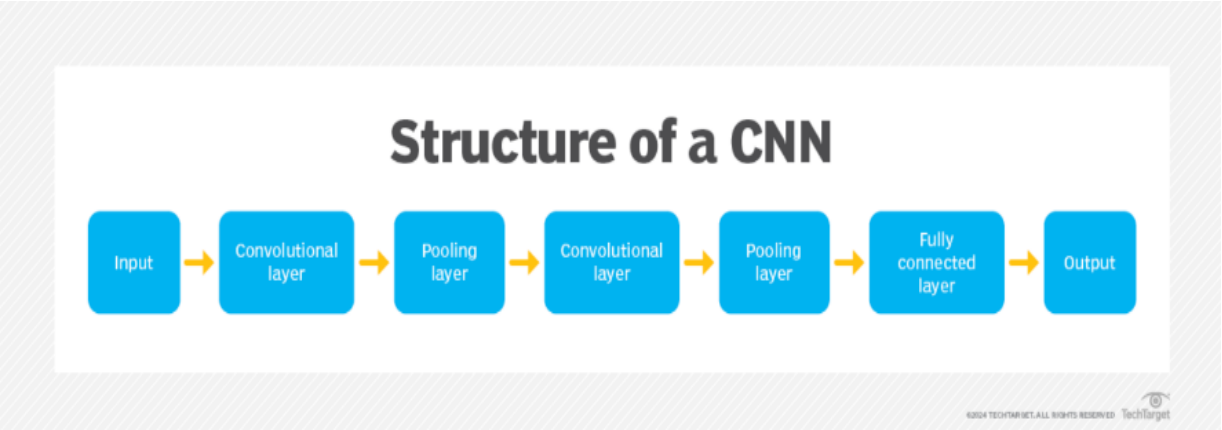


Figure 2.4: Structure of CNN.

Convolutional Layer

The convolutional layer is the core component of a CNN, where most computations take place. It applies small matrices called filters or kernels to an image, scanning its width and height to identify distinctive features. At each position, the filter computes a dot product between its weights and the corresponding pixel values, generating feature maps. These feature maps highlight specific patterns such as edges, textures, or shapes. As the network deepens, successive convolutional layers detect increasingly complex features, enabling the model to understand intricate patterns and objects within an image [29].

Pooling Layer

Following the convolutional layer, the pooling layer reduces the dimensionality of the feature maps while preserving essential information. This enhances the efficiency of the network and prevents overfitting. The most common pooling technique is max pooling, which selects the highest pixel value within a defined region, while average pooling computes the mean value of the region. By downsampling the data, the pooling layer helps CNNs generalize better while maintaining critical spatial characteristics [29].

Fully Connected Layer

The fully connected layer is responsible for final classification. Every neuron in this layer is linked to all neurons from the preceding layer, allowing the network to analyze all extracted features collectively. This layer maps the detected patterns to predefined categories, determining the final classification output. However, excessive fully connected layers can lead

to overfitting and increase computational costs, so CNNs often limit their number for efficiency and improved generalization [29].

Additional Layers

- **Activation Layer**: Introduces non-linearity to the network, enabling it to learn complex patterns. Common activation functions include ReLU, Sigmoid, Softmax, and Tanh. This layer is usually placed after convolutional or fully connected layers [29].
- **Dropout Layer**: Helps prevent overfitting by randomly disabling neurons during training, reducing model complexity and improving generalization [29].

4.5.2 Advantages of CNNs in Deep Learning

Deep learning, a subset of machine learning, leverages multi-layered neural networks to achieve superior performance compared to traditional single-layer models. Convolutional Neural Networks (CNNs) provide several advantages in deep learning applications, particularly in image processing and pattern recognition [29].

- **Highly Effective in Computer Vision**: CNNs are specifically designed for computer vision tasks such as image classification and object recognition. Their layered architecture enables them to identify fundamental features in early stages and progressively capture complex patterns in deeper layers, making them highly efficient in analyzing visual data.
- **Automated Feature Extraction**: A key strength of CNNs is their ability to automatically extract features from input images without requiring manual intervention. This eliminates the need for traditional feature engineering, which is often a labor-intensive and complex process.
- **Adaptability Through Transfer Learning**: CNN models can be repurposed for different tasks using transfer learning, where a pre-trained network is fine-tuned for new applications. This reusability makes CNNs particularly advantageous for scenarios with limited training data, reducing the need for extensive retraining while improving efficiency.
- **Computational Efficiency**: Unlike fully connected networks, CNNs utilize parameter sharing, significantly reducing the number of required computations. This makes them well-suited for deployment on various platforms, including mobile devices and edge computing environments, where processing power and memory are limited.

4.5.3 Limitations of CNNs

Despite their powerful capabilities, Convolutional Neural Networks (CNNs) come with certain challenges that can affect their efficiency and usability in different applications [29].

- **High Computational Cost**: Training CNN models requires substantial computational power, often necessitating specialized hardware such as GPUs or TPUs. Additionally, fine-tuning hyperparameters can be complex and time consuming.
- **Large Data Requirements**: CNNs typically need vast amounts of labeled data to achieve optimal performance. Collecting and annotating such datasets can be labor-intensive and costly, making it challenging for tasks with limited data availability.
- **Risk of Overfitting**: CNNs may become prone to overfitting, particularly when dealing with small datasets. Without techniques such as dropout layers or data augmentation, the model may memorize training data rather than generalizing well to new inputs.

4.5.4 Applications of Convolutional Neural Networks (CNNs)

Due to their ability to process and analyze visual data efficiently, CNNs have found widespread applications across various industries, including healthcare, automotive, social media, and retail. Their versatility extends beyond image processing, contributing to advancements in several real-world domains [29]:

- **Medical Diagnostics**: In healthcare, CNNs play a crucial role in analyzing medical images such as X-rays, MRI scans, and pathology slides. By identifying abnormalities and patterns indicative of diseases, CNNs assist medical professionals in diagnosing conditions more accurately and efficiently.
- **Autonomous Vehicles**: The automotive industry leverages CNNs for self-driving technology, enabling vehicles to interpret camera and sensor data to detect obstacles, traffic signs, and pedestrians. Additionally, CNNs enhance driver-assistance systems in traditional vehicles, improving features such as adaptive cruise control and automated parking.
- **Social Media Analysis**: Social media platforms utilize CNNs for various image-related tasks, such as automatic tagging suggestions, facial recognition, and content moderation. These models help enhance user experience by recommending relevant content and ensuring compliance with platform guidelines.

- **Retail and E-commerce**: Online retailers integrate CNNs into visual search systems, allowing customers to find products by uploading images instead of relying on text-based searches. Furthermore, CNNs enhance recommendation engines by identifying visually similar products, improving personalized shopping experiences.
- **Voice and Audio Processing**: Although primarily designed for image-based tasks, CNNs have also been applied in speech recognition systems. They assist virtual assistants in identifying spoken keywords and processing user commands, improving the accuracy and responsiveness of AI-driven voice interfaces.

4.6 Transfer Learning

Transfer learning refers to a technique in machine learning where knowledge gained from solving one problem is applied to a different but related problem. Instead of training a model from scratch, an existing model trained on a large dataset is adapted to a new task, which is especially helpful when labeled data for the new task is limited. This method enhances efficiency by reusing previously learned features, leading to faster convergence and improved results. Additionally, it helps reduce overfitting since the model begins with broadly applicable features that have already been optimized [30].

4.6.1 Importance of Transfer Learning

Transfer learning addresses several common challenges [30]:

- **Data scarcity**: Collecting and labeling large datasets can be difficult and expensive. By leveraging models that have already been trained on large datasets, transfer learning reduces the need for extensive new data.
- **Improved accuracy**: Since pre-trained models already possess learned features from broad datasets, they can achieve high performance more quickly and accurately when adapted to new tasks.
- **Efficiency in time and cost**: Utilizing pre-existing models significantly cuts down the time and computational cost compared to training a model from scratch.
- **Flexibility**: Pre-trained models can be fine-tuned to work effectively on related problems, making this approach highly adaptable across different domains, such as image classification and language processing.

4.6.2 Advantages of Transfer Learning

- **Faster model training**: Since the model has already been trained on a related task, it can adapt more quickly to the new task, requiring less time to learn relevant patterns and features [30].
- **Improved results**: Transfer learning often boosts the model's effectiveness by utilizing previously acquired knowledge, which helps achieve better accuracy on the new problem [30].
- **Effective with limited data**: In cases where the new task lacks sufficient data, transfer learning is especially useful. The model begins with a strong foundation of generalized features, which helps reduce the likelihood of overfitting [30].

4.6.3 Common pre-trained models

Numerous pre-trained models are widely used in image classification tasks due to their robust performance and proven reliability across different applications.

ResNet (Residual Networks)

Introduced by Microsoft Research, ResNet brought a major advancement in deep learning by incorporating residual connections, which help overcome issues like the vanishing gradient in very deep networks. The model comes in various configurations such as ResNet-50, ResNet-101, and ResNet-152, featuring deep architectures with shortcut connections that facilitate gradient flow. ResNet is frequently used for tasks like image classification, object detection, and feature extraction [31]. An overview of the ResNet architecture is illustrated in (figure 2.5).

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 2.5: ResNet architecture [32]

VGG (Visual Geometry Group Network)

Created by the Visual Geometry Group at the University of Oxford, VGG networks are recognized for their straightforward structure and depth. The common versions, VGG-16 and VGG-19, use a consistent 3x3 convolution filter design. VGG is commonly applied to image classification problems and serves as a strong feature extractor in many pipelines [31]. A representation of the VGG16 architecture is shown in (Figure 2.6).

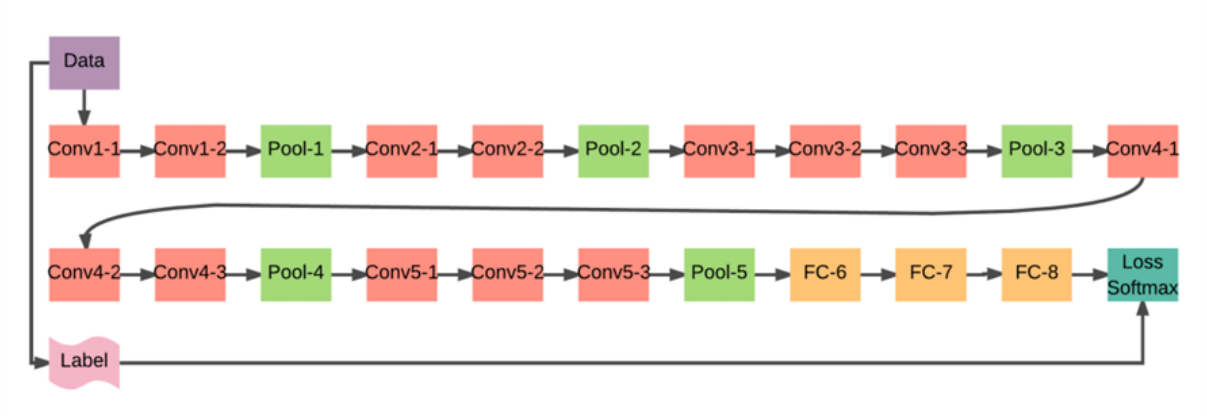


Figure 2.6: VGG16 architecture [33]

EfficientNet

Developed by Google, EfficientNet models are known for achieving high accuracy with significantly fewer parameters compared to traditional architectures. They use a compound scaling technique that uniformly scales the network's depth, width, and resolution. Variants range from EfficientNet-B0 to B7, making them ideal for both performance and efficiency in classification and transfer learning tasks [31]. An illustration of the EfficientNet architecture is provided in (Figure 2.7).

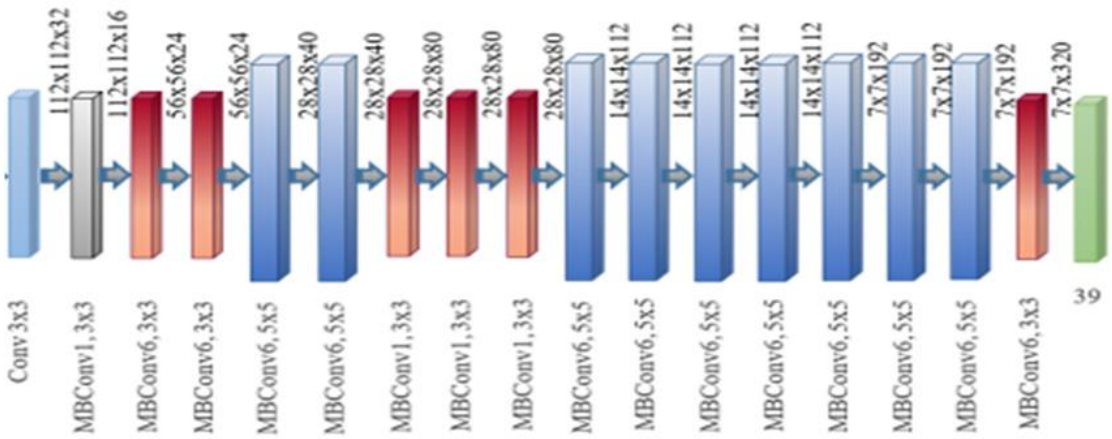


Figure 2.7: EfficientNet architecture [34]

DenseNet (Densely Connected Convolutional Networks)

Proposed by researchers from Cornell University, DenseNet models differ by connecting each layer to every other layer in a feed-forward manner. This dense connectivity encourages feature reuse and improves gradient flow, resulting in fewer parameters compared to standard convolutional networks. Variants include DenseNet-121, DenseNet-169, and DenseNet-201, commonly used in classification and feature extraction tasks [31]. A visualization of the DenseNet architecture is shown in (Figure 2.8).

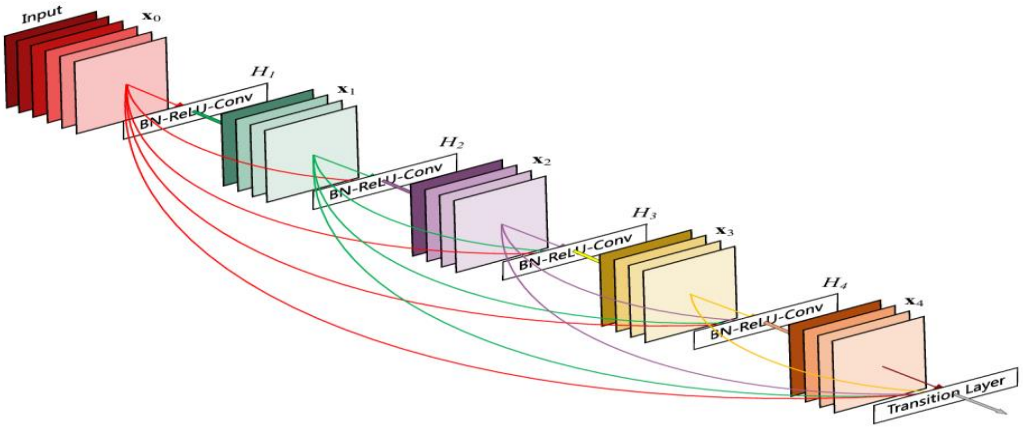


Figure 2.8: DenseNet architecture [35]

MobileNet

Designed by Google, MobileNet models are optimized for performance on mobile and embedded devices. With lightweight architecture and the use of depthwise separable convolutions, MobileNet comes in versions like V1, V2, and V3. These models are particularly useful for mobile-based vision systems and applications requiring low computational resources [31]. An illustration of the MobileNet architecture is presented in (Figure 2.9).

Type/Stride	Filter Shape	Input Size
Conv / s2	3 × 3 × 3 × 32	224 × 224 × 3
Conv dw / s1	3 × 3 × 32 dw	112 × 112 × 32
Conv / s1	1× 1 × 32 × 64	112 × 112 × 32
Conv dw / s2	3 × 3 × 64 dw	112 × 112 × 64
Conv / s1	1 × 1 × 64 × 128	56 × 56 × 64
Conv dw / s1	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 128	56 × 56 × 128
Conv dw / s2	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 256	28 × 28 × 128
Conv dw / s1	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 256	28 × 28 × 256
Conv dw / s2	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 512	14 × 14 × 256
Conv dw / s1 5 ×	3 × 3 × 512 dw	14 × 14 × 512
Conv / s1	1 × 1 × 256 × 512	14 × 14 × 512
Conv dw / s2	3 × 3 × 512 dw	14 × 14 × 512
Conv / s1	1 × 1 × 512 × 1024	7 × 7 × 512
Conv dw / s2	3 × 3 × 1024 dw	7 × 7 × 1024
Conv / s1	1 × 1 × 1024 × 1024	7 × 7 × 1024
Avg Pool / s1	Pool 7 × 7	7 × 7 × 1024
FC/ s1	1024 × 1000	1× 1 × 1024
Softmax / s1	Classifier	1× 1 × 1000

Figure 2.9: MobileNET architecture [36]

4.7 Related works

In recent years, the application of artificial intelligence (AI) and deep learning (DL) techniques in plant disease detection has gained significant attention. Various research studies have explored different machine learning (ML) and deep learning models to enhance the accuracy and efficiency of disease identification. The primary focus of these studies has been on improving model performance using datasets, preprocessing techniques, and advanced neural network architectures.

Tejaswini et al. [37] proposes a CNN-based model for early detection of plant diseases in tomato, potato, and bell pepper leaves. The model consists of three convolutional blocks (32, 64, and 128 filters) followed by fully connected layers, trained on 20,639 images from the Plant Village dataset. Achieving 86.21% accuracy, it effectively classifies 12 distinct diseases.

In another contribution, Shanthi et al. [38] proposes a CNN-based system for detecting tomato leaf diseases, comparing custom CNN, AlexNet, and VGG-16 architectures. Trained on 10 classes from the Plant Village dataset, their custom CNN achieved 91.2% overall accuracy, with model size (1.5MB). VGG-16 outperforming AlexNet in most disease categories (86% accuracy for Bacterial Spot).

The study by Najim et al. [39] proposes a CNN model for detecting 10 tomato leaf diseases, achieving 96% accuracy on the PlantVillage dataset. Their architecture uses only 4 layers (3 convolutional + 1 fully connected), significantly reducing computational costs compared to deeper models like ResNet50 (50 layers) while maintaining competitive performance. Achieving a model size (23MB)

Pandian et al. [40] proposes a 14-layer DCNN model for detecting 42 diseases across 16 plant species, achieving 99.97% accuracy the highest reported in literature. Their model combines synthetic data augmentation (DCGAN, NST, and BIM) to generate 147,500 images, addressing class imbalance. The 14-DCNN architecture, optimized via random search hyperparameter tuning, outperforms transfer learning models (VGG16, ResNet) in accuracy.

The study by Ali et al. [41] proposes an ensemble of four deep learning models (EfficientNetB0, InceptionResNetV2, DenseNet201, EfficientNetB3) for plant disease classification, achieving 99.89% accuracy the highest among ensemble-based approaches. Their novel CLAHE+AMF image pre-processing and class-weighted data balancing address challenges of noise and dataset imbalance in the 38-class PlantVillage dataset.

The study by Hamim and Jony [42] developed an optimized MobileNet model for chilli disease detection, achieving 97.18% accuracy a 6.53% improvement over the base MobileNet. Their architecture modifications included a reduced width multiplier (α=0.75), added dropout (0.5), and enhanced dense layers.

Finally, Alsayed et al. [43] proposes a deep learning framework for classifying diseases in apple tree leaves using transfer learning. Four pre-trained models (VGG16, ResNetV2, InceptionV3, MobileNetV2) were evaluated on the Plant Pathology 2020-FGVC7 dataset

(1,821 images of healthy leaves and three disease classes: apple scab, cedar apple rust, and multiple diseases). ResNetV2 with the Adam optimizer achieved the highest accuracy (94.7%), outperforming other models and published methods.

Table 2.4: Comparative Table of Recent Research.

Article (Ref)	Year	Dataset Name	Type of plant	Images / Classes	Model	Accuracy
[37]	2024	PlantVillage	Tomato, Potato, Bell pepper	20,639 images, 15 classes	Custom CNN	86.21%
[38]	2024	PlantVillage	Tomato	10 classes	Custom CNN, AlexNet, VGG-16	Custom CNN: 91.2%
[39]	2024	PlantVillage	Tomato	11,000 images, 10 classes	Custom CNN	96%
[40]	2022	Combined datasets	16 Plant crops	147,500 images, 58 classes	14-layer DCNN	99.97%
[41]	2024	PlantVillage	14 Plant crops	87,000 images, 38 classes	Ensemble (EfficientNetB0 + InceptionResNet V2 + DenseNet201+ EfficientNetB3)	99.89%
[42]	2024	Custom Chilli dataset	Chilli	300 images, 4 classes	Modified MobileNet	97.18%
[43]	2021	Plant Pathology 2020-FGVC7	Apple	1821 images 4 classes	VGG16, ResNetV2, InceptionV3, MobileNetV2	ResNetV2 94.7%

The table above (Table 2.4) summarizes the various studies examined as part of our study. These studies represent the latest research using deep learning for plant disease detection, particularly those that use the PlantVillage dataset.

Discussion

The reviewed articles demonstrate a wide range of deep learning approaches applied to plant disease detection, reflecting the growing interest in leveraging CNNs and transfer learning for agricultural applications. While methodologies and targeted crops vary, a common objective among these studies is to develop accurate, efficient models.

Most studies make use of the PlantVillage dataset, a publicly available and extensively used source for plant disease images, either in full or by selecting specific crops [41], [39]. A smaller number utilize custom or private datasets.

Several works focus on multi-species disease detection using large datasets and complex architectures. For example, Tejaswini et al. [37], Pandian et al. [40], and Ali et al. [41] explore broad disease classification tasks across different plant species using CNNs, ensembles, or deep architectures, achieving high accuracy. Other studies take a crop-specific approach, concentrating on tomato, apple or chili leaf diseases (Shanthi et al. [38], Najim et al. [39], Hamim and Jony [42], Alsayed et al [43]). These works often use either subsets of the PlantVillage dataset or custom datasets collected specifically for their study.

Although DCNN-type (Deep Convolutional Neural Network) architectures and model ensembles such as those proposed by Pandian et al. [40] and Ali et al. [41] achieve high accuracy, their computational complexity and large memory requirements often make them unsuitable for deployment on lightweight or mobile devices. This issue is particularly relevant given that most studies rely on the PlantVillage dataset and do not consider real-world constraints. Only a few works propose lightweight models optimized for practical use, and even fewer specifically address apple leaf diseases a gap that our research aims to fill.

Although existing studies show high performance, they often suffer from a lack of generalization, mainly due to the exclusive use of controlled datasets like PlantVillage. Few studies validate their models on real-world data.

Our work contributes to the ongoing development of practical deep learning solutions for agriculture by focusing specifically on apple leaf disease diagnosis.

The objective is to identify a model that can be realistically deployed in resource-constrained environments, such as mobile devices used directly in the field. By doing so, we aim to support the early diagnosis of plant diseases and provide accessible, practical tools for farmers and agricultural practitioners.

4.8 Conclusion

This chapter explored AI fundamentals, image recognition, focusing on deep learning and convolutional neural networks (CNNs). We examined CNN architecture, its benefits and challenges, and its applications, also transfer learning models. Additionally, a review of state-of-the-art research highlighted advancements in the field. This foundation sets the stage for the next chapters of Experimental Protocol and Results and development of web application.

Chapter 03
Experimental Protocol and Results

6.1 Introduction

This chapter describes the experimental protocol, implementation details, and results of the apple leaf disease diagnostic models developed in this study. The main goal was to classify apple leaf images into different categories healthy or affected by specific diseases using deep learning techniques. To accomplish this, three pretrained models ResNet18, MobileNetV2, and EfficientNetB0 were fine-tuned through transfer learning. Additionally, a custom Convolutional Neural Network (CNN) was built from scratch.

The chapter covers the hardware and software environment, the adopted methodology, and the data preparation process using the PlantVillage dataset. It also presents the model design, training settings, and evaluation metrics, followed by the obtained results. To support interpretability, Grad-CAM is applied on a small set of real-world leaf images to visualize the model's focus areas and better understand how decisions are made. The chapter concludes with a discussion of model's effectiveness, offering useful insights into the use of deep learning for apple leaf disease diagnosis. A final section illustrates the development of the best-performing model through a user-friendly web interface, enabling real-time interaction with the system.

6.2 Hardware and Software Environments

This section describes the hardware and software setup used during the implementation of the deep learning models. It includes the programming tools, libraries, and computing platform employed for development, training, and evaluation.

3.2.1 Tools and frameworks

Python 3.10.12

Python is a high-level, interpreted programming language known for its simplicity and readability. It supports multiple programming paradigms, including object-oriented, imperative, and functional programming. Designed to be highly extensible and versatile, Python enables developers to build applications efficiently, making it widely used in various fields such as data science, artificial intelligence, web development, and automation. Its extensive standard library and active community further contribute to its popularity and ease of use [44].



Figure 3.1: Python logo.

Pythorch

PyTorch is an open-source machine learning framework designed for rapid development and deployment of deep learning models. It offers dynamic computation graphs, high flexibility, and seamless integration with Python, making it intuitive and efficient for both research and production [45].



Figure 3.2: Pytorch logo.

NumPy

NumPy is a core Python library for scientific computing, offering efficient multidimensional array objects and a wide range of tools for performing mathematical, logical, and statistical operations, as well as linear algebra, data manipulation, and simulations [46].



Figure 3.3: Numpy logo.

Pandas

Pandas is an open-source Python library designed for efficient data manipulation and analysis, providing flexible data structures and functions to handle and process structured data with ease [47].

Matplotlib

Matplotlib is an open-source Python library widely used for creating a wide range of static, animated, and interactive visualizations. Introduced by John D. Hunter in 2003, it allows users

to transform raw data into clear and informative plots, making data analysis more intuitive and accessible [48].



Figure 3.4: Matplotlib logo [49].

Scikit-learn

Scikit-learn is a widely used Python library that offers efficient tools for machine learning and statistical modeling, including classification, regression, clustering, and dimensionality reduction. Built on NumPy, SciPy, and Matplotlib, it provides a simple and consistent interface for various machine learning tasks [50].



Figure 3.5: Scikit-learn logo.

Torchvision

Torchvision is a library in the PyTorch ecosystem, offering pre-trained models, datasets, and tools for image processing and augmentation. It's ideal for tasks like image classification, object detection, and segmentation, and integrates seamlessly with PyTorch for deep learning applications [51].

Kaggle

Kaggle is an online platform for data science and machine learning practitioners, where users can collaborate, access and share datasets, utilize GPU-powered notebooks, and participate in competitions to solve various data science problems [52].



Figure 3.6: Kaggle logo.

3.2.2 Hardware Configuration

Platform

For this project, we used the Kaggle platform, a cloud-based environment designed for data science and machine learning. Kaggle provides integrated Jupyter notebooks with access to limited computational resources, including GPUs, making it suitable for efficiently training deep learning models within certain constraints.

GPU

For training the models, we utilized Kaggle's GPU environment equipped with an NVIDIA Tesla P100-PCIE GPU. This GPU, with 16 GB of dedicated memory, provided significant acceleration for training convolutional neural networks by handling complex computations efficiently. However, despite its strong performance, the GPU's memory and computational resources are limited compared to higher-end hardware

Storage

The dataset was stored directly in Kaggle's cloud storage, specifically under the /kaggle/input/plantvillage-dataset directory. Kaggle's cloud-based file system provided seamless and fast access to the dataset, enabling efficient data loading and processing throughout the experiments.

6.3 Methodology

The methodology adopted in this study follows a structured pipeline (figure 3.7) designed to classify apple leaf images into four categories: Apple Scab, Black Rot, Cedar Apple Rust, and Healthy. The process begins with the use of the PlantVillage dataset, specifically the apple diseases subset. In the preprocessing step, all images were resized to 224×224 pixels, and data augmentation techniques were applied to improve generalization. The model design phase included both pretrained architectures (ResNet18, MobileNetV2, and EfficientNetB0) using transfer learning, and a custom CNN developed from scratch. During training, each model was trained for 20 epochs using a batch size of 32 and the Adam optimizer. Finally, the trained models were evaluated based on standard metrics including accuracy, precision, recall, F1-score, and confusion matrix to assess their classification performance.

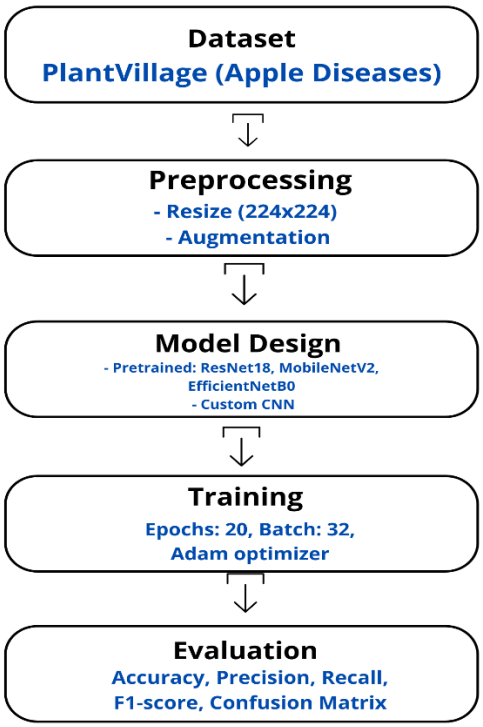


Figure 3.7: Workflow of the Plant Disease Detection Methodology.

6.4 Data Preparation

The quality and relevance of data are crucial factors influencing the performance of deep learning models, particularly in image classification tasks. In this work, the data preparation phase plays a central role. It consists of two main stages: the selection and utilization of the PlantVillage dataset, renowned for its richness and reliability in the field of plant disease detection and the preprocessing of images to ensure compatibility with the requirements of convolutional models. To further enhance the diversity of training examples and improve model robustness, a data augmentation strategy was also implemented. The following subsections provide a detailed overview of these key steps.

6.4.1 Data Collection

In this study, the dataset used for training and evaluating the deep learning models is the PlantVillage dataset, a publicly available and widely used benchmark dataset in the field of plant disease diagnosis [53]. It was developed by researchers at Penn State University to support the application of artificial intelligence in agriculture, specifically for identifying plant diseases through image classification.

The PlantVillage dataset contains over 54,000 labeled images of healthy and diseased plant leaves collected under controlled conditions. These images represent 38 distinct classes, covering various disease types and healthy conditions across 14 different crop species, including [53]:

- Apple
- Blueberry
- Cherry
- Corn (maize)
- Grape
- Orange
- Pepper
- Raspberry
- Soybean
- Squash
- Strawberry
- Tomato
- Peach
- Potato

The figure 3.8 shows some samples from PlantVillage dataset [54].

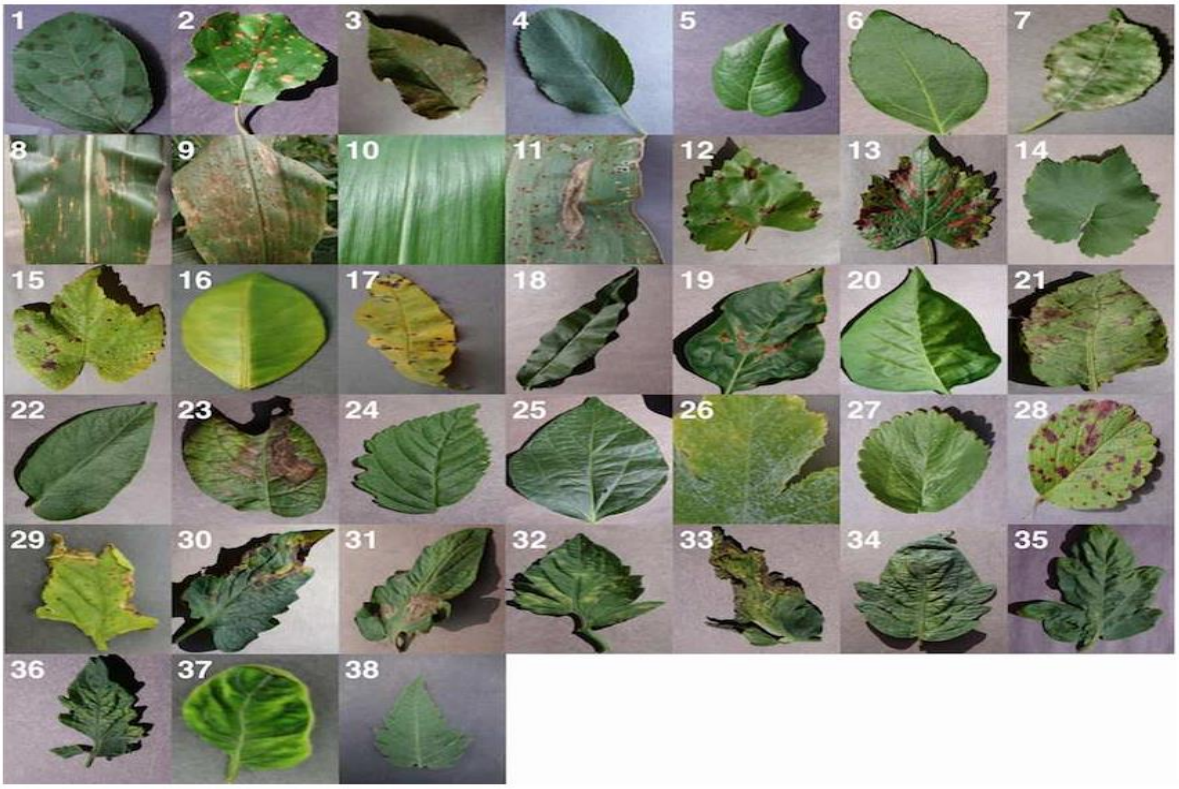


Figure 3.8: Sample Leaf Images Extracted from the PlantVillage Dataset.

Each image in the dataset is labeled according to the plant species and the specific disease (or healthy status), making it well-suited for supervised learning tasks.

For the purpose of this thesis, a focused subset of the dataset was selected to study apple leaf diseases, due to the apple tree's economic and agricultural importance. This subset includes four distinct classes, with approximately 3171 images in total. Each class represents either a specific fungal infection or a healthy condition of apple leaves, distributed as follows at table 3.1:

Table 3.1: Classes of apple diseases.

Class	Number of Images
Apple___Apple_scab	630
Apple___Black_rot	621
Apple___Cedar_apple_rust	275
Apple___healthy	1645
Total	**3171**

For effective evaluation of the models, we divided the dataset into three subsets based on a widely accepted ratio in supervised learning tasks 70% for training, 15% for validation, and 15% for testing. This division yielded approximately 2,219 images for the training set, 475 images for the validation set, and 477 images for the test set.

6.4.2 Data preprocessing and augmentation strategy

Image preprocessing is a crucial step in preparing data for deep learning models, especially in computer vision tasks such as plant disease classification. It improves the quality and consistency of the input data, and enhances significant features. This, in turn, leads to better model performance, faster convergence during training, and increased robustness to unseen samples.

Preprocessing Pipeline

For this study, the preprocessing pipeline consisted of the following steps:

- **Image Resizing**: All images were resized to 224 × 224 pixels, which is the standard input size for most CNN-based models, including ResNet18, MobileNetV2, and EfficientNetB0.
- **Conversion to Tensors**: Images were converted to tensors using PyTorch's ToTensor() function, which scales pixel values from the original 0–255 range to a normalized range of [0, 1]. This standardization ensures consistent input intensity and helps stabilize the model's learning process.

Data Augmentation

To improve the generalization capability of the Custom CNN model, real-time data augmentation techniques were applied to the training set only. These transformations increased the variability of the training data, helping the model learn more robust and invariant features. The augmentation techniques used were:

- **Random Horizontal Flip (p = 0.5):** This transformation mirrors images along the vertical axis, helping the model become invariant to different leaf orientations.
- **Random Vertical Flip (p = 0.3):** This adds further variability by flipping images along the horizontal axis, simulating different viewpoints.

- **Random Rotation (up to 45 degrees):** This introduces spatial variation, mimicking natural changes in leaf positioning or camera angles.
- **Color Jitter (brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1):** This simulates changes in lighting and color conditions, making the model more robust to visual inconsistencies.

These augmentations were applied only during training, not to validation or test sets, to avoid data leakage and ensure consistent and unbiased evaluation. To illustrate the data augmentation effects.

To illustrate the data augmentation effects, we captured sample images after applying the transformations to the training data. Examples in the figure 3.9 show how augmentation increases the variability of the dataset and helps improve the model's robustness.



Figure 3.9: Samples of augmented images.

6.5 Model Design and Training

To effectively classify plant diseases from leaf images, it was essential to design and train models capable of learning relevant visual features. This section presents the model architectures used in the study, including both pretrained networks and a custom built CNN.

6.5.1 Pretrained Models

In the context of this study, we integrated three well-established and high-performing convolutional neural network (CNN) architectures: **ResNet18**, **MobileNetV2**, and **EfficientNetB0**. These models were carefully selected due to their computational efficiency, and their proven accuracy in image classification tasks, particularly on large-scale datasets such as ImageNet.

The architectural and performance characteristics of the selected models are summarized in Table (He et al., 2016 [55]; Sandler et al., 2018 [56]; Tan and Le, 2019 [57]; Simonyan and Zisserman, 2015 [58]). This table compares the key attributes of common CNN architectures in terms of number of parameters, model size and their accuracy on ImageNet.

Table 3.2: Comparative Characteristics of Selected CNN Architectures.

Model	Params (Millions)	Size (MB)	Accuracy (ImageNet)
ResNet18	11.7	45	69.8%
MobileNetV2	3.5	14	72.0%
EfficientNetB0	5.3	20	77.1%
VGG16 (excluded)	138	528	71.5%

The choice of ResNet18, MobileNetV2, and EfficientNetB0 was guided by their favorable trade-off between accuracy and resource efficiency (table 3.2). Other models like VGG16 were excluded due to their high parameter count, making them unsuitable for real-time or mobile deployment.

To adapt these pretrained models to the specific task of plant disease diagnosis, we employed a transfer learning approach. Transfer learning is a widely used technique in deep learning, where a model trained on a large and diverse dataset is repurposed for a different, yet related, task. In our case, all three models were initially pretrained on ImageNet, a benchmark dataset that contains over one million labeled images across 1000 object categories. Leveraging the rich feature representations already learned by these models enabled us to benefit from their deep visual understanding while reducing the training time and mitigating overfitting especially given that my dataset was relatively limited in size.

To implement this strategy, we performed the following modifications across all three architectures:

- First, we froze all the convolutional layers, thereby preserving the pretrained weights and feature extractors. This means that during training, these layers remained unchanged and continued to act as fixed feature extractors.
- Next, we replaced the final classification layer (which was originally tailored for 1000 ImageNet classes) with a new fully connected layer designed specifically for our classification task.
- This new final layer outputs four neurons, each corresponding to one of the disease categories in our dataset: Apple Scab, Black Rot, Cedar Apple Rust, and Healthy.
- Only this final layer was trained from scratch on our dataset, allowing the models to learn how to map the extracted features to my specific labels.

6.5.2 Proposed Model (Custom CNN)

In parallel with the use of pretrained models, we designed a custom Convolutional Neural Network (CNN) (figure 3.10) specifically tailored for the task of classifying apple leaf images into one of four categories: Apple Scab, Black Rot, Cedar Apple Rust, or Healthy.

The model begins with four convolutional layers, each using a 3×3 kernel. The number of filters increases progressively 16, 32, 64, and 128 allowing the network to extract increasingly complex and high-level features from the input images. Each convolutional layer is followed by a Rectified Linear Unit (ReLU) activation function to introduce non-linearity, and a max pooling operation with a 2×2 window to reduce spatial dimensions and computational cost while preserving essential features.

After the final convolutional block, an adaptive average pooling layer is employed to compress the spatial resolution of feature maps to 1×1, ensuring flexibility in handling varying image sizes. The resulting tensor is flattened and passed through a fully connected layer with dropout (set at 0.3). The final classification is performed by a fully connected output layer composed of four neurons, each representing one of the target classes, performs the final classification. We apply a SoftMax activation function to produce probability distributions over the classes.

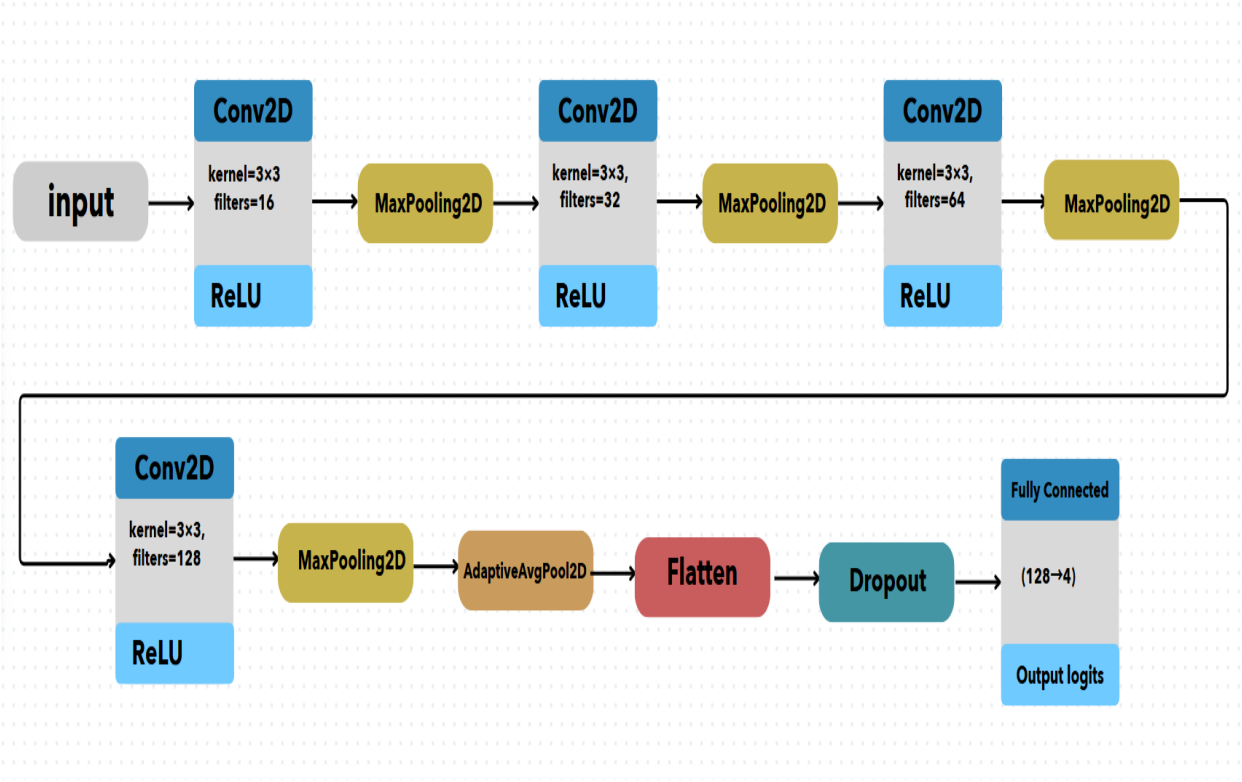


Figure 3.10: Custom CNN Architecture Diagram.

6.5.3 Models Training

Hyperparameter optimization is a crucial step in the deep learning process, as it directly influences the convergence speed, model stability, and generalization ability. In this work, manual tuning was performed based on the observed performance on the validation set, particularly accuracy and loss function. The tested hyperparameters include the learning rate, batch size, dropout rate, choice of optimizer, and number of epochs. The following (table 3.3) summarizes the different values explored and the selected configurations for the final experiments.

Table 3.3: Hyperparameter search space and final selected configuration.

Hyperparameter	Values Tried	Final Value
Learning Rate	0.1, 0.01, 0.001	0.001
Batch Size	16, 32, 64	32
Dropout Rate	0.2, 0.3, 0.5	0.3
Optimizer	Adam, SGD	Adam
Epochs	10, 20, 30	20

All experiments were conducted on Kaggle's cloud-based Jupyter environment, which provides access to a Tesla P100 GPU. The entire implementation was developed using the Python programming language, with PyTorch serving as the primary deep learning framework due to its flexibility, dynamic computation graphs, and strong support for transfer learning and model customization. These controlled training settings allow for reliable performance evaluation across the different model architectures used in this study.

6.6 Evaluation

This section presents the evaluation phase of the implemented models, which is essential to assess their effectiveness in classifying apple leaf diseases. Standard performance metrics and comparative analysis are used to measure both the accuracy and practicality of each model.

6.6.1 Performance metrics

This section presents the evaluation metrics used to measure the performance of the developed models. Key indicators such as accuracy, precision, recall, and F1-score are discussed, along with the confusion matrix for visualizing classification results.

Confusion Matrix

The confusion matrix is a table that summarizes the model's performance (figure 3.11) by comparing actual and predicted labels. It includes the following components [59]:

- **TP (True Positives)**: Correctly predicted positive cases.
- **FP (False Positives)**: Incorrectly predicted positive cases.
- **TN (True Negatives)**: Correctly predicted negative cases.
- **FN (False Negatives)**: Incorrectly predicted negative cases.

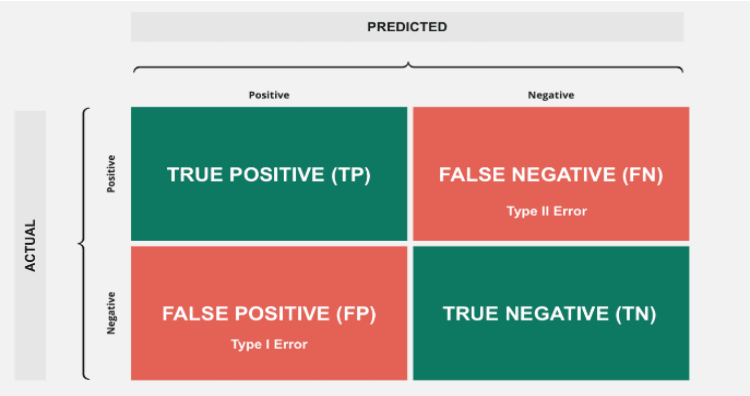


Figure 3.11: Representation of Confusion-matrix [60].

Accuracy

Accuracy represents the ratio of correctly predicted observations to the total observations. It gives a general measure of how often the model makes correct predictions [59].

Formula:

$$\text{Accuracy} = (TP+TN) / (FP+FN+TP+TN)$$

Precision

Precision is a performance metric that measures the ratio of correctly predicted positive instances to the total predicted positives. It evaluates the model's ability to avoid false positives and is particularly useful when the cost of false alarms is high [59].

Formula:

$$\text{Precision} = TP / (TP + FP)$$

Recall

Recall (also known as sensitivity or true positive rate) measures the ratio of correctly predicted positive instances to all actual positives. It assesses the model's ability to detect all relevant cases, especially important when missing a true case is critical [59].

Formula:

$$\text{Recall} = TP / (TP + FN)$$

F1-Score

The F1-score is the harmonic mean of precision and recall. It provides a balanced measure that takes both false positives and false negatives into account. It is especially useful when the class distribution is imbalanced [59].

Formula:

F1-score = 2 * (Precision * Recall) / (Precision + Recall)

6.6.2 Obtained results

In this study, a comprehensive evaluation of the four implemented deep learning models is presented. The evaluation is based on key performance metrics including on multiple performance indicators, including training, validation, and test accuracy, the best-performing epoch as well as model size and structural complexity. These criteria allow for a balanced assessment that considers not only predictive performance but also practical considerations such as computational cost and deployment feasibility.

The summarized comparison of the models is shown in Table 3.4.

Table 3.4: Summarized comparison of all models.

Model	Test Accuracy	Train Accuracy	Validation Accuracy	Validation epochs	Model Size	Parameters
ResNet18	0.99	0.9950	0.9895	epoch 17	45 MB	11.7 millions
MobileNetV2	0.99	0.9937	0.9916	Epoch 18	14 MB	3.5 millions
EfficientNetB0	0.99	0.9901	0.9874	Epochs 12	20 MB	5.3 millions
Custom CNN	0.98	0.9865	0.9895	Epoch 15	**0.37 MB**	**97,956**

All four models evaluated in this study (**ResNet18, MobileNetV2, EfficientNetB0**, and the **Custom CNN**) demonstrated strong classification capabilities, achieving test accuracies of 98% or higher. While the table presents only accuracy metrics, all models also recorded F1-scores, precision, and recall values exceeding 0.98, confirming the robustness of their performance across multiple evaluation criteria. These high scores reflect the models' ability to

effectively balance false positives and false negatives, which is crucial in the context of plant disease detection. This underscores the effectiveness of deep learning, particularly CNNs and transfer learning, in accurately identifying plant diseases.

However, a closer examination of the performance indicators reveals notable distinctions. MobileNetV2, designed for lightweight applications, delivered one of the best balances: with a test accuracy of 99%, reaching its peak at epoch 18, while being significantly more compact at 14 MB and 3.5 million parameters. EfficientNetB0 and ResNet18 followed closely with high test accuracy of 99%. Nevertheless, both models possess significantly larger sizes and parameter counts compared to MobileNetV2.

ResNet18 achieved a high test accuracy of 99%, with its best validation performance reached at epoch 17. While its performance is excellent, the model requires 45 MB of storage and has a parameter count of 11.7 million, making it relatively heavy and computationally expensive. Similarly, EfficientNetB0 reached 99% test accuracy, with a validation accuracy of 98.74%, peaking at epoch 12**.** Though slightly more efficient than ResNet18 in terms of size (20 MB) and parameters (5.3 million), it still demands a considerable amount of memory and processing power. These pretrained models exhibit high classification performance but require significant memory resources due to their size.

The **Custom CNN model**, on the other hand, emerges as the most efficient and practical architecture among all evaluated models. With only 97,956 parameters and a storage of 0.37 MB, it is remarkably lightweight it still achieved a test accuracy of 98%, with its best performance recorded at epoch 15. This means it performs well on new data while using much less memory than the pretrained models. Although its test accuracy is lower than the other models, the difference is negligible (just 1%). This makes the Custom CNN a strategic choice for real-time applications, such as mobile devices, embedded systems.

The figure 3.12 presents the classification report of our Custom CNN model, detailing its performance across all four classes. This includes precision, recall, F1-score, and support values for each class.

	precision	recall	f1-score	support
Apple___Apple_scab	1.00	0.92	0.96	91
Apple___Black_rot	0.99	1.00	1.00	104
Apple___Cedar_apple_rust	1.00	1.00	1.00	37
Apple___healthy	0.97	1.00	0.98	245

Figure 3.12: Classification report of the CustomCNN.

Based on the classification report shown above, the Custom CNN model demonstrates strong and reliable performance across all classes. Its high accuracy and F1-scores reflect the model's effectiveness in distinguishing between healthy and diseased apple leaves.

The model showed outstanding performance in classifying **Black Rot**, with 0.99 precision, 1.00 recall, and a perfect F1-score, correctly diagnosing all real cases with almost no false predictions. For **Cedar Apple Rust**, it achieved flawless scores (1.00 for precision, recall, and F1-score), proving its strong ability to recognize the disease's unique visual patterns without confusion.

For the **Healthy** class, the model achieved 1.00 recall and 0.97 precision, indicating all healthy leaves, with slight misclassification of diseased samples. The F1-score of 0.98 confirms strong overall performance. For **Apple Scab**, it reached 1.00 precision, 0.92 recall, and a 0.96 F1-score, meaning all predicted cases were correct, though a few real cases were missed likely due to visual similarity with healthy or related leaves.

To better assess the performance of the Custom CNN, the confusion matrix in (figure 3.13), illustrates the performance of the model across the four classes. The model achieves perfect classification for Black_rot (104/104) and Cedar_apple_rust (37/37), indicating its strong ability to classify these diseases. It also performs very well on the Healthy class, correctly classifying 244 out of 245 samples, with only one misclassified as Black_rot.

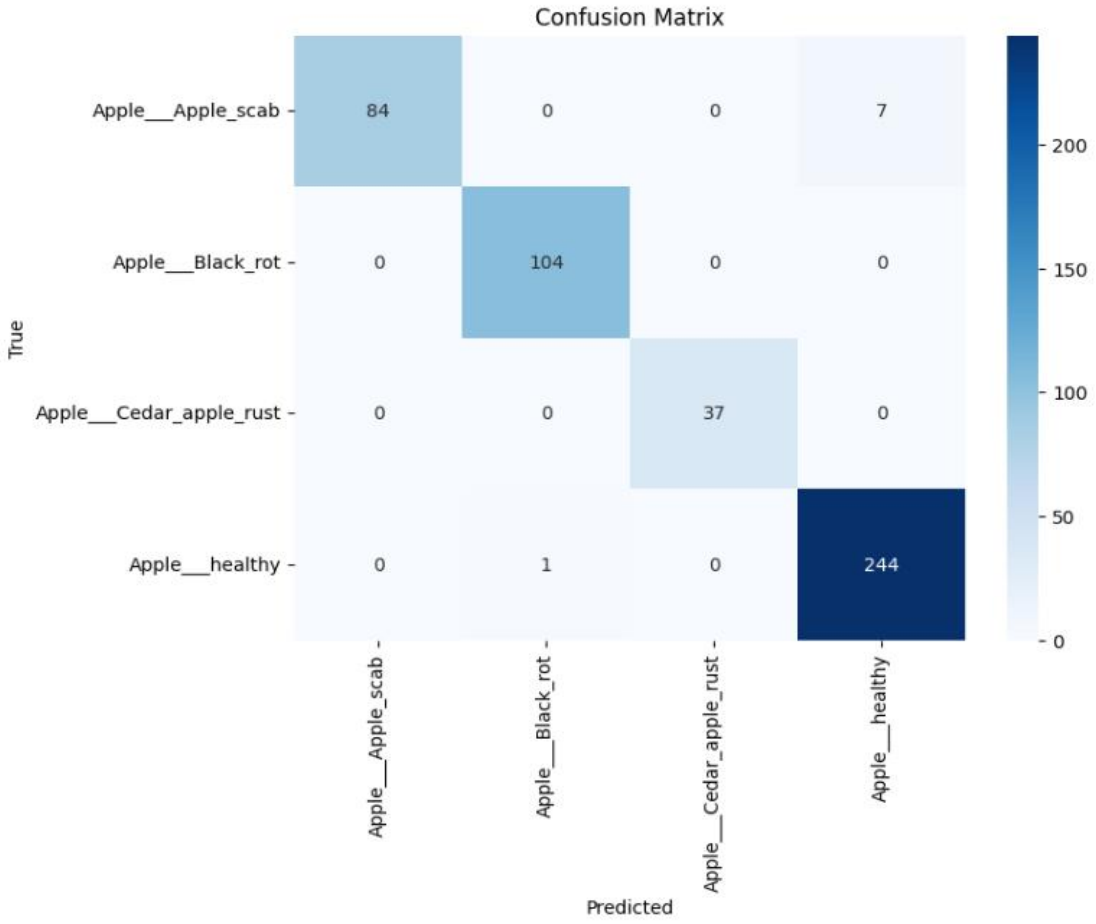


Figure 3.13: Confusion Matrix (CustomCNN).

Overall, the CustomCNN model demonstrates robust classification performance, especially in classifying Black_rot and Cedar_apple_rust with 100% accuracy, while maintaining high accuracy for the remaining classes. Examples of correct predictions for Cedar_apple_rust and Black_rot are shown in (figure 3.14).

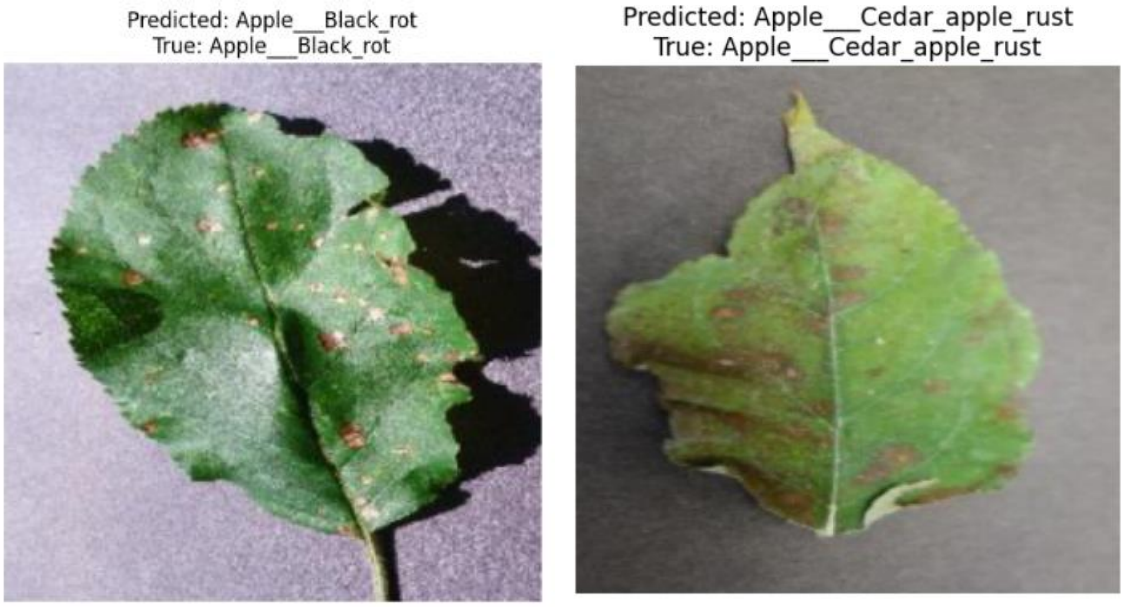


Figure 3.14: Examples of correctly labeled apple leaves.

However, the model shows slight confusion between Apple_scab and Healthy, with 7 Apple_scab samples incorrectly predicted as Healthy (figure 3.15). This confusion is likely due to visual similarities between early Apple_scab symptoms and healthy leaves.



Figure 3.15: Example of misclassified image.

Discussion

All three pretrained models ResNet18, MobileNetV2, and EfficientNetB0 achieved a high accuracy of 99. MobileNetV2 stood out by combining strong performance with model size (14 MB) and 3.5 million parameters. ResNet18 also delivered competitive performance but had a significantly larger model size (45 MB) and 11.7 million parameters, making it less suitable for low-resource environments. EfficientNetB0 offered a balanced performance with (20 MB and 5.3 million parameters).

In contrast, the Custom CNN architecture emerged as the most balanced and promising model across all key evaluation criteria. Despite being designed from scratch without the benefit of transfer learning or pretrained weights, the model achieved a highly competitive test accuracy of 98%, while maintaining an extremely lightweight footprint only 97,956 trainable parameters and a model size of 0.37 MB.

What distinguishes the Custom CNN developed is its efficiency without compromising accuracy. It demonstrates that with task specific design and effective architectural choices, compact convolutional neural networks can achieve performance levels comparable to pretrained models. This compact yet high performing structure is especially advantageous for real time and embedded applications, such as disease diagnostic on mobile devices, drones, or edge devices deployed in agricultural environments.

Our custom model, although effective (98% accuracy), remains inferior to MobileNetV2. This can be explained by its simpler architecture and a more limited number of layers. A possible improvement would be to add residual layers or an attention mechanism.

6.7 Exploratory evaluation of the model on real-world images

To evaluate the performance of our model and understand its internal workings, we conducted a series of exploratory tests on a small sample of real apple leaf images, captured outside the original PlantVillage dataset. These images include natural variations in lighting conditions, background noise, and orientation, allowing us to assess the model's robustness in a more realistic context and better understand its decision-making mechanisms using the Grad-CAM (Gradient-weighted Class Activation Mapping) tool.

Grad-CAM is a neural network interpretability method that generates a heatmap overlaid on the input image. This technique makes it possible to visualize the regions of an image that were most influential in the model's decision. By highlighting the activated areas during prediction, it provides an effective way to analyze the relevance of the convolutional neural network's reasoning, especially in sensitive contexts such as plant disease detection [61].

6.7.1 Experimental Protocol

In this testing phase, five image pairs were selected to represent a variety of cases: correct predictions (of healthy or diseased leaves), as well as an example of a misclassification. The images used come from the PlantDoc dataset, an open-source database containing real images of plant leaves captured in natural conditions, including variations in lighting, background, and orientation [62]. Unlike synthetic or laboratory controlled datasets, PlantDoc better reflects the visual complexity encountered in the field, making it a relevant choice for evaluating a model's robustness in realistic scenarios. This dataset includes multiple plant species and common diseases and was published by Singh et al. (2020) as a resource for research in computer vision applied to agriculture.

For each image, we observed the model's prediction, compared it to the actual label, and then generated a Grad-CAM visualization superimposed on the original image. These visualizations allowed us to identify the most influential regions in the model's decision-making process, using a color scale ranging from blue (less relevant areas) to red (highly contributive regions).

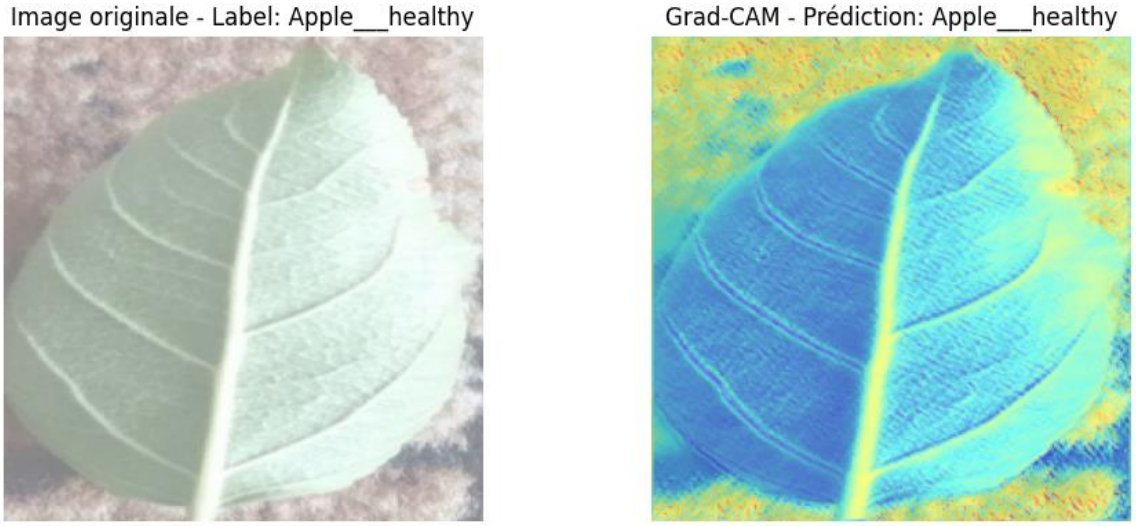


Figure 3.16: Case 1 – Healthy Leaf (Original Image and Grad-CAM Map).

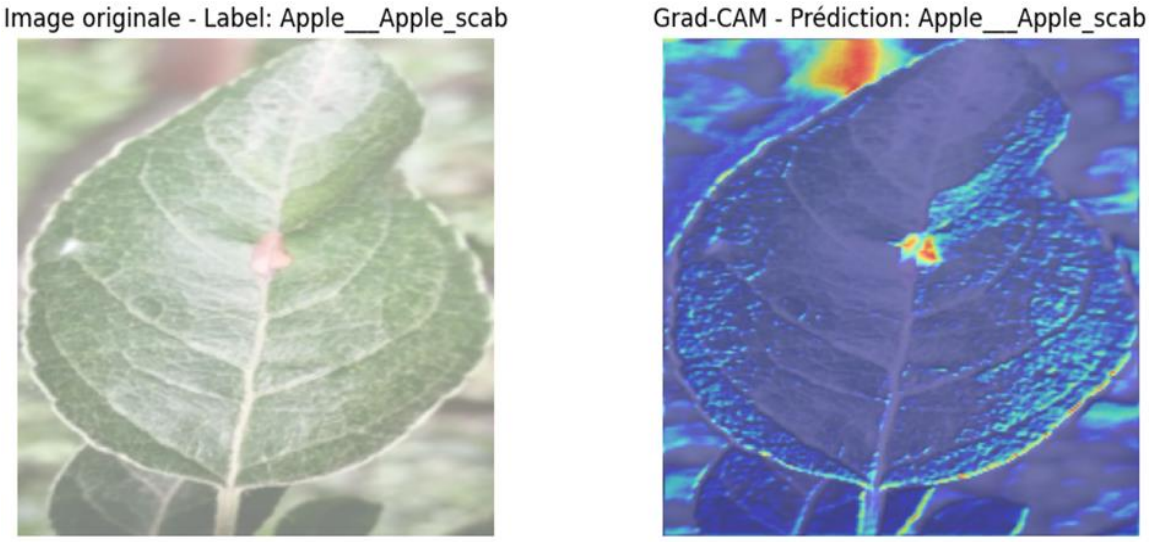


Figure 3.17: Case 2 - Apple Scab (Visible Lesion : Original Image and Grad-CAM).

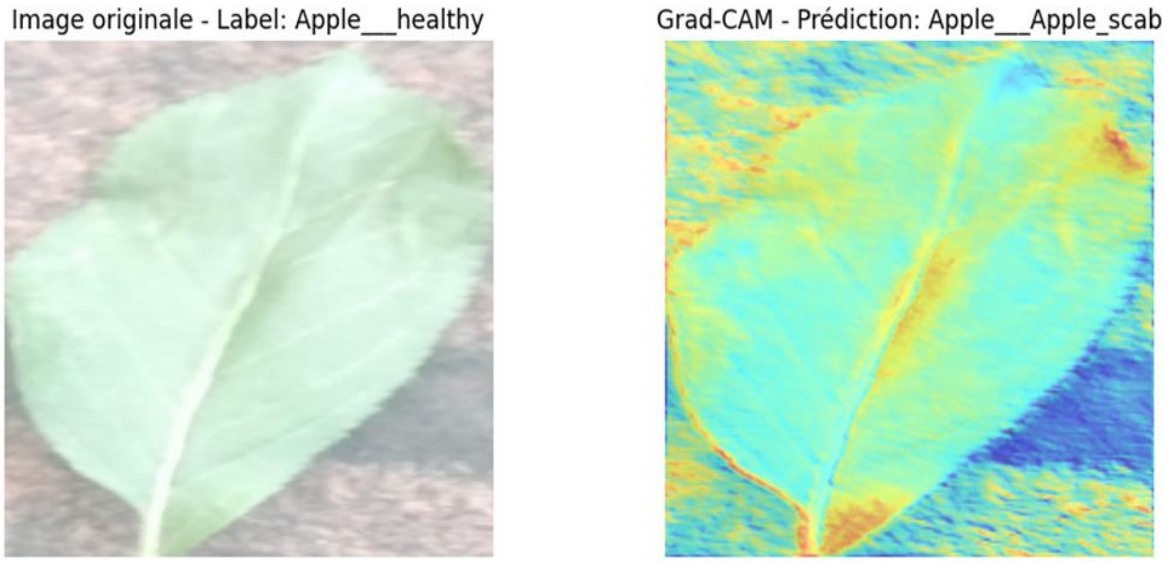


Figure 3.18: Case 3-Healthy Image Predicted as Diseased (Original Image and Grad-CAM Map).

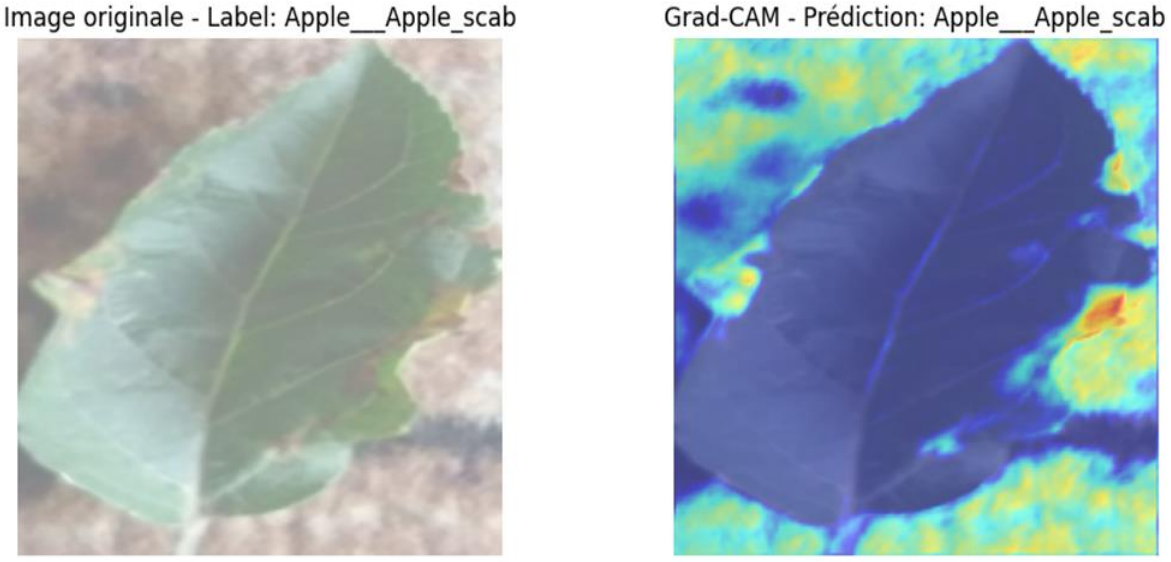


Figure 3.19: Case 4 - Apple Scab (Original Image and Grad-CAM).

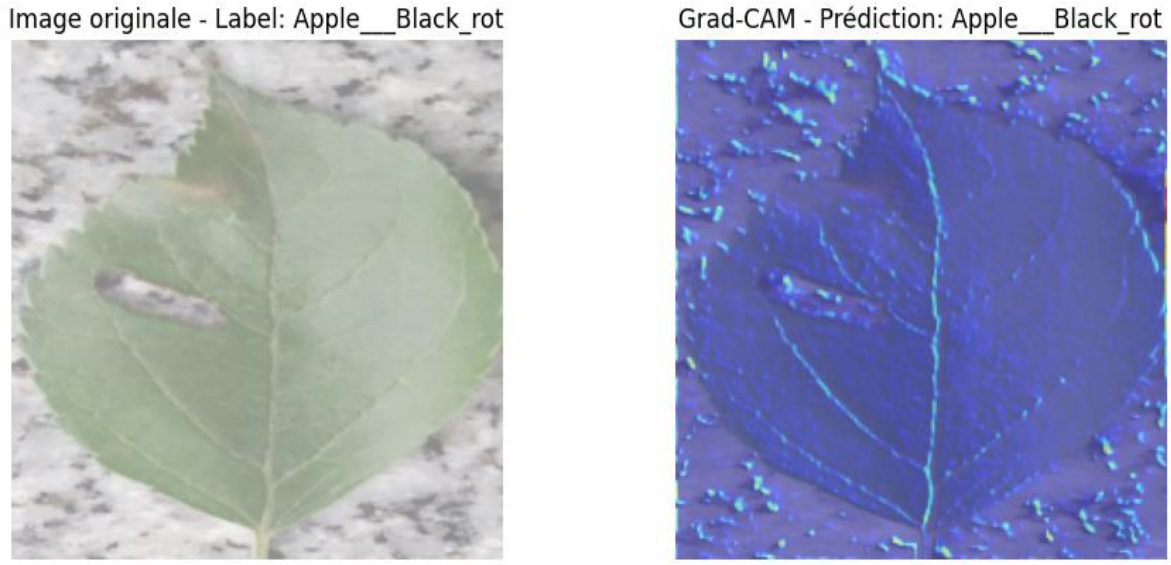


Figure 3.20: Case 5 - Black Rot (Original Image and Grad-CAM).

6.7.2 Qualitative Analysis of Predictions Using Grad-CAM

In the **first case** the image shows a perfectly healthy leaf with no visible symptoms (Figure 3.16). The model correctly predicts the *Apple__healthy* class, and the Grad-CAM analysis indicates a focus on the central veins and edges of the leaf, demonstrating a good recognition of normal plant structures.

The **second case** concerns a leaf affected by *Apple_scab*, with a clearly visible dark lesion (Figure 3.17). The model correctly predicts the class, and Grad-CAM shows a clear activation centered on the lesion. This result demonstrates that the model has learned to detect the visual signs of this fungal disease.

The **third case** highlights a misclassification: a healthy leaf is incorrectly predicted as Apple_scab (Figure 3.18). The Grad-CAM activation is focused on a shadow area near the leaf margin, rather than on any pathological features. This indicates that the model mistook the shadow for a disease symptom, suggesting a sensitivity to visual artifacts like lighting variations or shadows. Improving robustness to such perturbations will be essential for real-world deployment. For example, the dataset can be diversified by incorporating real-world images captured under natural conditions (varying lighting).

In the **fourth case** the model correctly identifies a leaf affected by *Apple_scab*, even though the symptoms are more diffuse (Figure 3.19). Grad-CAM activations are distributed around the veins and slightly discolored areas. This ability to detect more subtle signs confirms a certain robustness of the model when faced with less obvious manifestations of the disease.

Finally, in the **fifth case** a leaf with a necrotic lesion characteristic of *Apple_Black_rot* is correctly classified (Figure 3.20). Grad-CAM highlights targeted activation around the lesion and main veins. Although the overall intensity of activations is moderate, the relevant localization of the regions of interest suggests a reliable behavior of the model in this case.

The following table summarizes the tests performed (Table 3.5).

Table 3.5: Summary of Model Predictions and Grad-CAM Interpretations.

Image	True Class	Model Prediction	Activated Regions (Grad-CAM)	Interpretation
1	Apple__healthy	Apple__healthy	Veins, edges	Good recognition of a healthy leaf
2	Apple__Apple_scab	Apple__Apple_scab	Central lesion (Dark spot)	Relevant focus on the symptom
3	Apple__healthy	Apple__Apple_scab	Veins and edges without anomalies	Confusion between natural structures and disease
4	Apple__Apple_scab	Apple__Apple_scab	Diffuse areas (Discoloration)	Good detection despite subtle symptoms
5	Apple__Black_rot	Apple__Black_rot	Necrotic area, main veins	Correct visual interpretation with moderate confidence

Discussion

The visual analysis of the predictions using Grad-CAM made it possible to validate several crucial aspects regarding the model's behavior. In most cases, the decisions are based on relevant visual features that are well localized and consistent with the biological reality of the symptoms. The method also helped identify generalization errors, particularly in a false positive case, thereby highlighting the importance of diversity in the training data and environmental conditions.

This experiment confirms that our model is capable of producing visually interpretable predictions on the tested real-world images. However, it also reveals weaknesses when faced with visual elements not represented in the training data.

The use of Grad-CAM not only enhances the model's transparency but also guides future improvement directions, including:

- Enriching the dataset with images captured under real-world conditions,
- Implementing robustness techniques against natural visual disturbances.

These insights shape our future perspective: to systematically validate the model on real-world data, in a mobile or field application context, in order to overcome the methodological limitations of existing approaches.

After evaluating the model's robustness on real-world data, a web application was developed to allow its practical use in a user-friendly environment.

6.8 Application Development

The final objective of our work is to propose a web application for diagnosing apple diseases based on a deep learning model. This section presents the Application development phase of our system through a web application. We first introduce the tools and frameworks used to develop the interface. Second the backend of the application. Then we describe the design, functionalities, and testing of the implemented model.

6.8.1 Tools and Frameworks Used

The Application development phase of the system focused on designing a user-friendly and interactive interface that allows real-time diagnosing of apple leaf diseases using a Custom CNN model. To achieve this, a combination of specialized tools and libraries was employed, ensuring a seamless integration between the trained model and the user interface. This section outlines the technologies adopted during development, along with the rationale behind their selection.

Streamlit

We selected **Streamlit**, a Python-based open-source framework, to build the web application interface. Streamlit enabled us to design an interactive and responsive user interface, this framework integrates naturally with PyTorch, allowing us to load the trained model, process user-uploaded images, and display the result in real time [63].

We favored Streamlit for its rapid development capabilities, and strong compatibility with machine learning models, which made it an ideal choice for this project.

PyTorch and Torchvision

We used **PyTorch** to train the convolutional neural network (CNN) and maintained the same framework to ensure model compatibility and reduce development complexity [45]. PyTorch allowed us to efficiently load the trained model weights.

We also utilized **Torchvision**, a companion library to PyTorch, to perform critical image preprocessing steps [51]. These included resizing the input images, normalizing pixel values, and converting images to tensors, transformations required to ensure input compatibility with the trained CNN model.

Pillow (PIL)

We used the **Pillow** library (also known as PIL) to handle image input. Pillow allowed us to open and convert user-uploaded images to the RGB format, which ensured consistency in preprocessing. Its speed and simplicity made it a suitable choice for real-time applications like ours [64].

CSS

To enhance the aesthetics of the application, we customized the user interface using **CSS styles** directly within the Streamlit environment. We used CSS to style elements such as buttons and layout containers, giving the application a modern and visually appealing design [65].

Visual Studio Code (VS Code)

We developed the entire application using **Visual Studio Code (VS Code)**, a powerful and widely used source code editor. It offered us robust support for Python, integrated terminal access, and extensions for Streamlit development. These features allowed us to write, test, and debug the application efficiently, accelerating the development process and maintaining code quality [66].

Python

We developed the entire application using **Python**, a widely adopted programming language in the fields of data science and artificial intelligence. Python offers a rich ecosystem of libraries, ease of syntax, and strong community support, which makes it ideal for rapid prototyping and model deployment [44].

6.8.2 Backend of the application

The backend is responsible for handling the model loading, image preprocessing, and classification logic. It includes the following components :

- **Model Definition and Loading**

The CustomCNN class defines a convolutional neural network architecture trained for multi-class classification of apple leaf diseases. The model was trained using PyTorch and

saved using `torch.save()`. It is reloaded in the app using `load_state_dict()` for inference.

- **Image Preprocessing**

Images uploaded by users are preprocessed using the Torchvision library. Preprocessing steps include resizing to 224x224 pixels, and conversion to tensors to match the model's input requirements.

- **Image Upload Handling**

The backend enables users to upload images through the `st.file_uploader()` function. Uploaded files are temporarily stored using Streamlit's `session_state`, allowing further processing without re-uploading on each interaction. The uploaded image is converted to RGB format using the PIL library before classification.

- **Prediction Function**

The prediction logic uses the softmax function to calculate class probabilities. The predicted class and its confidence score are returned and displayed to the user.

Here's a summary of the backend files and logic:

- model/best_modelcnn.pth: Trained PyTorch model.
- CustomCNN: Class defining the CNN architecture.
- load_model(): Loads the pre-trained model.
- predict(): Preprocesses images and returns prediction results.

To run the application locally, we used the following command: `streamlit run app.py`

This opens the app in a local browser at `http://localhost:8501`.

6.8.3 Interface Design and Testing

We developed an interactive web interface to demonstrate how our trained deep learning model performs in real-world scenarios. This interface allows users to easily interact with the system and visualize the disease diagnosis results based on uploaded leaf images.

The application displays a single-page layout, designed using Streamlit, which presents a seamless workflow from image upload to result display. Users begin by uploading a leaf image using the intuitive upload component. The system accepts JPEG, JPG and PNG formats and immediately displays the selected image within the interface, offering visual feedback.

To launch our apple diseases diagnostic system, we executed the application locally by running the app deployment script. Specifically, we opened a terminal within Visual Studio Code and ran it.

Upon execution, the Streamlit server automatically launched the interface in a new browser tab. This allowed us to visualize the user interface and begin interaction with the model in real-time. Figure 3.21 shows the homepage of our application, displaying the initial layout before any image upload.



Figure 3.21: Homepage of our web interface.

Once users upload an image, the **"Classify"** button becomes active, triggering the model inference process. The system processes the image using the loaded model and outputs both the identified class and the corresponding confidence score (probability value ‘ranging from 0% to 100%’ that represents how certain the model is about its identified class. Higher values indicate greater certainty in the identified outcome).

We tested the system using various apple leaf images selected from the PlantVillage dataset. The figure (figure 3.22) illustrates some examples of the classification results displayed by the web application.

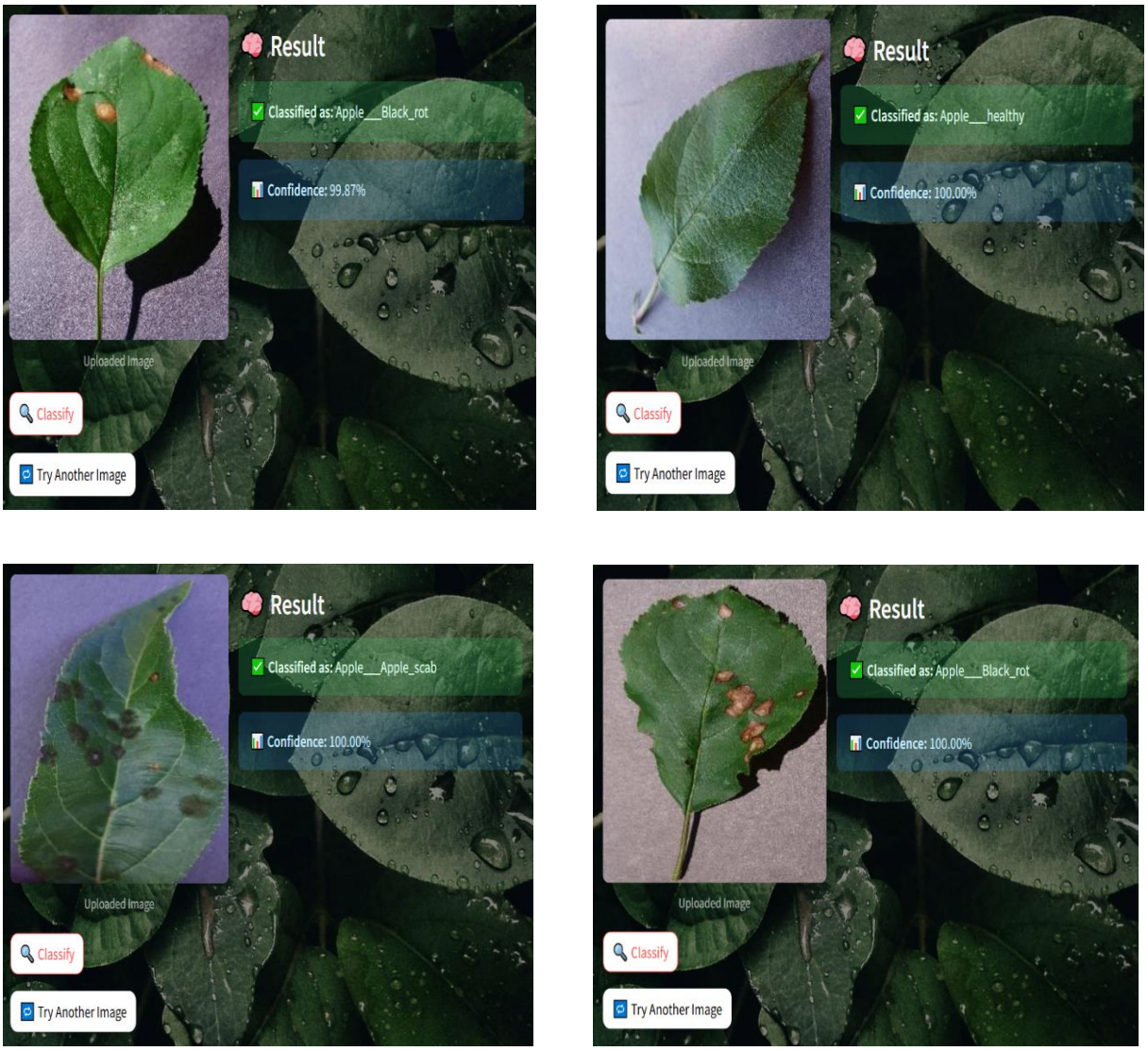


Figure 3.22: Examples Result for Different Apple Leaf Conditions.

6.9 Conclusion

This chapter presented the experimental setup, dataset preparation, model implementation, and evaluation of four deep learning models for plant disease classification. All models achieved high accuracy, while the Custom CNN proved to be a highly efficient lightweight solution. The results highlight the strong potential of deep learning, especially custom model, for real world agricultural applications. We also developed a web application to display the model's functionality in real-world scenarios, enabling real-time disease classification from leaf images. Furthermore, Grad-CAM was used to visualize and interpret the model's decisions, offering insight into the regions of the image that influenced each prediction.

General Conclusion and Future Work

General Conclusion and Future Work

The growing importance of agriculture in ensuring global food security makes early and accurate diagnostic of plant diseases more critical than ever. This thesis explored the application of artificial intelligence, specifically deep learning techniques, for the automated classification of plant diseases using image data.

The experiments were conducted on a subset of the PlantVillage dataset containing images of apple leaves, covering three types of diseases as well as healthy leaves. We evaluated the performance of three pretrained models ResNet18, MobileNetV2, and EfficientNetB0 along with a lightweight custom model designed as part of this work.

All models performed exceptionally well, achieving accuracy scores above 98%. However, these models are demanding in terms of computational power, hardware resources, and execution time. The custom CNN model stood out as a lightweight and resource-efficient solution, capable of delivering competitive results with minimal computational cost making it ideal for real-time deployment in low-resource environments such as rural farms or mobile applications.

In addition to the experimental evaluation, a practical implementation of the best-performing model was developed as a web application using Streamlit. This application enables users to upload images of apple leaves and receive real-time results regarding the presence and type of disease. The implementation demonstrates the feasibility and accessibility of deep learning based plant disease diagnosis through a user-friendly interface.

As part of the evaluation of our apple disease classification model, we used the interpretability method Grad-CAM (Gradient-weighted Class Activation Mapping). This method highlights the regions of the image that influenced the model's prediction by coloring the areas of interest according to their contribution to the decision.

It is important to recall that our classification model was initially trained on images from the PlantVillage database, a highly structured dataset captured under controlled conditions. The tests performed, although limited in number, were carried out on more varied and visually complex images, thus closer to real field conditions. This gap between the training and test data highlighted certain undesirable model behaviors while confirming its robustness in simple and moderately complex cases.

The tests conducted at this stage allowed for a detailed interpretation of the decisions made by our classification model. This approach therefore constitutes an exploratory step within a broader evaluation process on realistic data, aiming toward future deployment in an operational setting.

Perspectives

The experimental study conducted as part of this final project provides a foundation for future extensions of this work, oriented towards applicability in real agricultural contexts. Based on these observations, several avenues for improvement can be envisaged:

- **Diversification of the training dataset:** Integrate images collected under real-world conditions (visual noise, variable lighting, off-center angles) to reduce observed generalization errors.
- **Expanded validation on field data:** Conduct a large-scale testing campaign on images from real crops to assess the true performance of the model in an applied context.
- **Model improvement:** Introduce attention mechanisms or finer architectures to better discriminate natural textures from pathological symptoms.

Bibliography

[1] J. M. Alston and P. G. Pardey, "Agriculture in the global economy," *J. Econ. Perspect.*, vol. 28, no. 1, pp. 121-146, Feb. 2014, doi: 10.1257/jep.28.1.121.

[2] SEO Team, "Difference Between Traditional and Modern Farming," *Swasya Living*, Jul. 17, 2024. [Online]. Available: https://www.swasyaliving.com/post/difference-between-traditional-and-modern-farming

[3] National Institute of Food and Agriculture, "Plants," U.S. Department of Agriculture. [Online]. Available: https://www.nifa.usda.gov/topics/plants. [Accessed: May 12, 2025].

[4] M. M. Hossain *et al.*, "Plant disease dynamics in a changing climate: Impacts, molecular mechanisms, and climate-informed strategies for sustainable management," *Discover Agriculture*, vol. 2, p. 132, 2024, doi: 10.1007/s44279-024-00144-w.

[5] A. Schulman, "9 different types of plant fungal diseases," *Pest Control Options*, Jun. 13, 2023. [Online]. Available: https://pestcontroloptions.com/9-different-types-of-plant-fungal-diseases/. [Accessed: May 12, 2025].

[6] B. Hayes, "Bacterial leaf spot: How to identify and control this deadly plant disease," *MorningChores*, Apr. 1, 2021. [Online]. Available: https://morningchores.com/bacterial-leaf-spot/. [Accessed: May 12, 2025].

[7] S. Khan, "Bacterial wilt: Identify, prevent and organic treatment," *Envirevo Agritech*, Sep. 25, 2024. [Online]. Available: https://envirevoagritech.com/bacterial-wilt-treatment/. [Accessed: May 12, 2025].

[8] Agriculture Victoria, "Tomato yellow leaf curl virus," Department of Jobs, Precincts and Regions, Government of Victoria, May 12, 2024. [Online]. Available: https://agriculture.vic.gov.au/biosecurity/plant-diseases/vegetable-diseases/tomato-yellow-leaf-curl-virus. [Accessed: May 12, 2025].

[9] Family Handyman, "Most common plant diseases." [Online]. Available: https://www.familyhandyman.com/list/most-common-plant-diseases/. [Accessed: May 12, 2025].

[10] A. Jafar, N. Bibi, R. A. Naqvi, A. Sadeghi-Niaraki, and D. Jeong, "Revolutionizing agriculture with artificial intelligence: Plant disease detection methods, applications, and their limitations," *Front. Plant Sci.*, vol. 15, p. 1356260, 2024, doi: 10.3389/fpls.2024.1356260.

[11] K. A. Garrett, M. Nita, E. D. De Wolf, P. D. Esker, L. Gomez-Montano, and A. H. Sparks, “Plant pathogens as indicators of climate change,” in *Climate Change*, Elsevier, 2021, pp. 499–513.

[12] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in Plant Science*, vol. 7, p. 1419, 2016.

[13] New England Tree Fruit Management Guide, "Apple scab," University of Massachusetts Amherst. [Online]. Available: https://netreefruit.org/apples/diseases/apple-scab. [Accessed: May 12, 2025].

[14] H. Faubert, "Cedar-apple rusts," *New England Tree Fruit Management Guide*, University of Massachusetts Amherst. [Online]. Available: https://netreefruit.org/apples/diseases/cedar-apple-rusts. [Accessed: May 12, 2025].

[15] White Rot and Black Rot," *New England Tree Fruit Management Guide*, [Online]. Available: https://netreefruit.org/apples/diseases/white-rot-and-black-rot. [Accessed: May 20, 2025].

[16] M. Mohana, "The history of artificial intelligence," *Mohd Mohana*. [Online]. Available: https://www.mohdmohana.com/post/the-history-of-artificial-intelligence/. [Accessed: May 12, 2025]

[17] D. Harper, "Etymology of 'artificial'," *Online Etymology Dictionary*. [Online]. Available: https://www.etymonline.com/word/artificial?utm_. [Accessed: May 12, 2025].

[18] D. Harper, "Etymology of 'intelligence'," *Online Etymology Dictionary*. [Online]. Available: https://www.etymonline.com/word/intelligence. [Accessed: May 12, 2025].

[19] H. Sheikh, C. Prins, and E. Schrijvers, "Artificial intelligence: Definition and background," in *Mission AI*, Cham: Springer International Publishing, 2023, pp. 15–41. doi: 10.1007/978-3-031-21448-6_2.

[20] McKinsey & Company, "What is AI (artificial intelligence)?," *McKinsey Explainers*. [Online]. Available: https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-ai?utm_. [Accessed: May 12, 2025].

[21] I. Kash, "What is (AI) Artificial Intelligence?," *Online Master of Engineering*, University of Illinois Chicago, Dec. 21, 2023. [Online]. Available:

https://meng.uic.edu/news-stories/ai-artificial-intelligence-what-is-the-definition-of-ai-and-how-does-ai-work/. [Accessed: May 12, 2025].

[22] Z. Saleh, "Artificial Intelligence Definition, Ethics and Standards," Apr. 2019. [Online]. Available: https://www.researchgate.net/publication/332548325_Artificial_Intelligence_Definition_Ethics_and_Standards

[23] G. Guney *et al.*, "An overview of deep learning algorithms and their applications in neuropsychiatry," *Clin. Psychopharmacol. Neurosci.*, vol. 19, no. 2, pp. 206–219, May 2021, doi: 10.9758/cpn.2021.19.2.206.

[24] M. I. Perera, T. T. Nguyen, D. T. Nguyen, T. H. Trinh, L. T. Vo, and N. T. Nguyen, "A machine learning-based risk score for prediction of mechanical ventilation in children with dengue shock syndrome: A retrospective cohort study," *PLOS ONE*, vol. 19, no. 9, p. e0315281, Sep. 2024, doi: 10.1371/journal.pone.0315281.

[25] J. Devassy, A. Bharadwaj, and L. Kharb, *What is Image Recognition?*, 2019.

[26] Adobe, "Guide to image file formats," *Adobe*, [Online]. Available: https://www.adobe.com/acrobat/hub/guide-to-image-file-formats.html?utm. [Accessed: 12-May-2025].

[27] "Image classification vs detection vs segmentation," *Clarifai Docs*, [Online]. Available: https://docs.clarifai.com/tutorials/image-classification-detection-segmentation/?utm_. [Accessed: 12-May-2025].

[28] N. Klingler, "Image Segmentation with Deep Learning (Guide)," *viso.ai*, Sep. 25, 2023. [Online]. Available: https://viso.ai/deep-learning/image-segmentation-using-deep-learning/. [Accessed: 12-May-2025].

[29] A. S. Gillis, L. Craig, and R. Awati, "What is a convolutional neural network (CNN)?," *TechTarget*, Jan. 2025. [Online]. Available: https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network. [Accessed: 12-May-2025].

[30] GeeksforGeeks, "What is Transfer Learning?," *GeeksforGeeks*, Feb. 13, 2025. [Online]. Available: https://www.geeksforgeeks.org/ml-introduction-to-transfer-learning/. [Accessed: 12-May-2025].

[31] GeeksforGeeks, "Top Pre-Trained Models for Image Classification," *GeeksforGeeks*, Jul. 3, 2024. [Online]. Available: https://www.geeksforgeeks.org/top-pre-trained-models-for-image-classification/. [Accessed: 12-May-2025].

[32] D. Sarwinda, R. H. Paradisa, A. Bustamam, and P. Anggia, "Deep learning in image classification using residual network (ResNet) variants for detection of colorectal cancer," *Procedia Computer Science*, vol. 179, pp. 423–431, 2021.

[33] H. Qassim, A. Verma, and D. Feinzimer, "Compressed residual-VGG16 CNN model for big data places image recognition," in *Proc. 2018 IEEE 8th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Las Vegas, NV, USA, Jan. 2018, pp. 169–175.

[34] R. P. P. Fathima, A. Su, M. A. Moustafa, and M. A. Ali, "Detecting plant disease in corn leaf using EfficientNet architecture—An analytical approach," *Electronics*, vol. 12, no. 8, p. 1938, 2023.

[35] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 4700–4708.

[36] T. Ghosh, M. M. H. Z. Abedin, S. M. Chowdhury, Z. Tasnim, T. Karim, S. S. Reza, and M. A. Yousuf, "Bangla handwritten character recognition using MobileNet V1 architecture," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 6, pp. 2547–2554, 2020.

[37] P. Rastogi, S. Dua, and V. Dagar, "Early disease detection in plants using CNN," *Procedia Computer Science*, vol. 235, pp. 3468–3478, 2024.

[38] D. L. Shanthi, K. Vinutha, and N. Ashwini, "Tomato leaf disease detection using CNN," *Procedia Computer Science*, vol. 235, pp. 2975–2984, 2024.

[39] M. H. Najim, S. K. Abdulateef, and A. H. Alasadi, "Early detection of tomato leaf diseases based on deep learning techniques," *International Journal of Artificial Intelligence*, vol. 13, no. 1, pp. 509–515, 2024.

[40] J. A. Pandian, V. D. Kumar, O. Geman, M. Hnatiuc, M. Arif, and K. Kanchanadevi, "Plant disease detection using deep convolutional neural network," *Applied Sciences*, vol. 12, no. 14, p. 6982, 2022.

[41] A. H. Ali, A. Youssef, M. Abdelal, and M. A. Raja, "An ensemble of deep learning architectures for accurate plant disease classification," *Ecological Informatics*, vol. 81, p. 102618, 2024.

[42] S. A. Hamim and A. I. Jony, "Enhanced deep learning model architecture for plant disease detection in Chilli plants," *Journal of Edge Computing*, vol. 3, no. 2, pp. 136–146, 2024.

[43] A. Alsayed, A. Alsabei, and M. Arif, “Classification of apple tree leaves diseases using deep learning methods,” *International Journal of Computer Science & Network Security*, vol. 21, no. 7, pp. 324–330, 2021.

[44] Python Software Foundation, "What is Python? Executive Summary," *Python.org*, [Online]. Available: https://www.python.org/doc/essays/blurb/. [Accessed: 12-May-2025].

[45] PyTorch Foundation, "PyTorch Projects," *PyTorch*, [Online]. Available: https://pytorch.org/projects/pytorch/. [Accessed: 12-May-2025].

[46] NumPy Developers, "What is NumPy?," *NumPy Documentation*, [Online]. Available: https://numpy.org/doc/stable/user/whatisnumpy.html. [Accessed: 12-May-2025].

[47] GeeksforGeeks, "Introduction to Pandas in Python," *GeeksforGeeks*, May 12, 2025. [Online]. Available: https://www.geeksforgeeks.org/introduction-to-pandas-in-python/. [Accessed: 12-May-2025].

[48] GeeksforGeeks, "Introduction to Matplotlib," *GeeksforGeeks*, Feb. 18, 2025. [Online]. Available: https://www.geeksforgeeks.org/python-introduction-matplotlib/. [Accessed: 12-May-2025].

[49] Matplotlib Development Team, "Matplotlib: Visualization with Python," *Matplotlib*, [Online]. Available: https://matplotlib.org/. [Accessed: 12-May-2025].

[50] Tutorialspoint, "Scikit Learn Tutorial," *Tutorialspoint*, [Online]. Available: https://www.tutorialspoint.com/scikit_learn/index.htm. [Accessed: 12-May-2025].

[51] Torchvision Contributors, "Torchvision: Computer Vision Models, Datasets, and Transforms for PyTorch," *PyTorch*, [Online]. Available: https://docs.pytorch.org/vision/stable/index.html. [Accessed: 12-May-2025].

[52] DataCamp, "What is Kaggle?," *DataCamp*, [Online]. Available: https://www.datacamp.com/blog/what-is-kaggle. [Accessed: 12-May-2025].

[53] B. Wang, C. Zhang, Y. Li, C. Cao, D. Huang, and Y. Gong, "An ultra-lightweight efficient network for image-based plant disease and pest infection detection," *Precision Agriculture*, vol. 24, no. 5, pp. 1836–1861, 2023.

[54] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in Plant Science*, vol. 7, p. 1419, 2016.

[55] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770–778.

[56] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Salt Lake City, UT, USA, 2018, pp. 4510–4520.

[57] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, Long Beach, CA, USA, 2019.

[58] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Representations (ICLR)*, San Diego, CA, USA, 2015, pp. 1–14.

[59] Scikit-learn Developers, "Metrics and scoring: quantifying the quality of predictions," *Scikit-learn Documentation*, [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html. [Accessed: 12-May-2025].

[60] S. Polamuri, "Best Confusion Matrix Guide With Sklearn Python," *Dataaspirant*, Aug. 3, 2020. [Online]. Available: https://dataaspirant.com/confusion-matrix-sklearn-python/. [Accessed: 12-May-2025].

[61] K. Gopalan, S. Srinivasan, M. Singh, S. K. Mathivanan, and U. Moorthy, "Corn leaf disease diagnosis: Enhancing accuracy with ResNet152 and Grad-CAM for explainable AI," *BMC Plant Biol.*, vol. 25, no. 1, p. 440, 2025.

[62] D. Singh, N. Jain, P. Jain, P. Kayal, S. Kumawat, and N. Batra, "PlantDoc: A dataset for visual plant disease detection," in *Proc. 7th ACM IKDD CoDS and 25th COMAD*, 2020, pp. 249–253.

[63] J. Smith, "Create an Image Classification Web App using PyTorch and Streamlit," *Medium*, Feb. 3, 2025. [Online]. Available: https://medium.com/data-science/create-an-image-classification-web-app-using-pytorch-and-streamlit-f043ddf00c24

[64] S. Gruppetta, "Image Processing With the Python Pillow Library," *Real Python*, Jan. 8, 2025. [Online]. Available: https://realpython.com/image-processing-with-the-python-pillow-library/

[65] E. Dennis, "Beautify Streamlit With Custom CSS," *Medium*, 2024. [Online]. Available: https://medium.com/@ericdennis7/beautify-streamlit-with-custom-css-0500b44449cb

[66] M. Li, "VSCode-Python: The All-in-One Efficient Development Tool for Python Developers," *Medium*, Feb. 22, 2025. [Online]. Available: https://medium.com/top-

python-libraries/vscode-python-the-all-in-one-efficient-development-tool-for-python-developers-aa619d37dc90