Ministry of Higher Education and Scientific Research
Mohamed khider university of biskra
Faculty of Sciences and Technology
Department of Electrical Engineering

LMD Master Thesis

To obtain the LMD master's degree in telecommunication engineering

Plant Leaf Diseases using Convolutional Neural Network

By: **Yamouna MIMOUNE**

Publicly defended on: 02/05/2025

In front of the committee consisting of:

Academic Year 2024/2025

Dedition

In the name of love, obedience, and respect, I dedicate this work:

To my dear first teachers: my parents,

To my mother, a friend who gave me life, a symbol of kindness always ready to assure me that everything will be alright. To this mother, who wholeheartedly shares joys and sorrows with us,

To my father, this great man, always ready to lend a hand when I need it, this constant beacon guiding my steps, this wonderful father who never hesitated to sacrifice everything for his children,

Today, from the bottom of my heart, I say 'THANK YOU, my parents', the wellspring of courage,

To my dear brothers,

To my Dr, lovely sister 'DIDA'.,

To my bright beautiful aunt,

And to all my friends, both in my social and academic life.

Acknowledgment

I begin by expressing my gratitude to God Almighty for bestowing upon me this significant blessing and success on the path I have chosen. Without His grace, reaching this stage and completing it would not have been possible. I extend my sincere thanks to God for all His blessings. It is essential to acknowledge that the worldly means we employed were not the contributors to our progress; rather, it was the grace of God that served as the foundation for our success and accomplishment in this endeavor.

I would like to express my deepest gratitude and sincere appreciation to my thesis supervisor, Professor **Abdelmalik OUAMANE**, from the University of Biskra. His expert guidance, constant support, and patience throughout this research were instrumental to the success of this work. His valuable feedback and encouragement helped me stay focused and motivated during every stage of the project.

My sincere thanks go as well to Professor **Abdelkrim OUAFI**, from the University of Biskra, who honored me by presiding over my defense jury. I am truly grateful for his time, his interest in my work, and the insightful remarks he provided.

I also wish to express my warmest thanks to Dr.**Saadia MEDOUAKH**, from the University of Biskra, for her role as examiner. Her critical perspective and constructive comments have greatly enriched the quality of this research.

To my professors, jury members, and everyone who contributed directly or indirectly to this academic journey, I extend my heartfelt appreciation.

Undoubtedly, I cannot conclude without expressing my deepest gratitude to my parents. Their unwavering love, support, and sacrifices were vital to the achievement of this milestone. Their encouragement and belief in me were a constant source of strength.

Finally, I thank all those who, in one way or another, contributed to the realization of this work-through a smile, a word of encouragement, or a helping hand. Your kindness and support will never be forgotten.

Abstract

Agricultural productivity is facing growing threats from plant diseases, which cause an estimated $220 billion in losses each year and pose a serious risk to global food security. Traditionally, identifying these diseases has relied on manual inspections by experienced experts a process that's time consuming, labor-intensive, and prone to human error. To address these challenges, this study explores the use of deep learning-specifically, Convolutional Neural Networks (CNNs)-to automate the detection and classification of plant diseases.

While CNNs have shown great potential, their effectiveness is often limited by issues such as insufficiently diverse datasets, poorly optimized model configurations, and a lack of robustness in real-world conditions. In this research, we systematically evaluate several CNN architectures, including DenseNet121, AlexNet, and VGG19, using publicly available datasets. Our goal is to identify the best-performing models and configurations to improve detection accuracy.

By benchmarking these models and analyzing their key components, we offer practical insights and recommendations for applying deep learning in real agricultural environments. The results highlight the potential of intelligent monitoring systems to revolutionize crop health management, supporting more sustainable farming practices and strengthening global food security through advanced AI technologies.

Key words: Plant Diseases, Deep Learning, Convolutional Neural Networks (CNNs), Image Classification, DenseNet121, AlexNet, VGG19.

Table of Contents

List of Figures

General Introduction

Agriculture remains one of humanity's most vital endeavors, serving as the foundation that sustains our evergrowing global population. However, this essential sector faces mounting challenges, particularly from plant diseases that threaten agricultural productivity worldwide. According to the Food and Agriculture Organization of the United Nations (FAO), plant diseases cause staggering annual losses estimated at approximately *$220* billion globally, resulting not only in significant economic damage but also in compromised food security for millions [Nair, 2023].

Traditionally, managing these threats has relied heavily on the experienced eyes of farmers and agricultural experts who manually inspect crops for signs of disease. While this human expertise is invaluable, such approaches are inherently limited by their labor-intensive nature, time constraints, and susceptibility to human error. The subjective nature of visual inspection means earlystage diseases might go unnoticed, and subtle symptoms may be misinterpreted, potentially leading to inappropriate treatments that exacerbate crop damage. In response to these limitations, the agricultural community has increasingly embraced technological solutions. The emergence of machine learning and deep learning technologies offers unprecedented opportunities to revolutionize plant disease detection and classification. These advanced systems promise to automatically identify and classify plant diseases with remarkable accuracy, enabling more reliable, efficient, and scalable crop monitoring [Mohyuddin et al., 2024].

Among these technologies, Convolutional Neural Networks (CNNs) have dominated the landscape of plant disease detection, achieving impressive results in image recognition tasks. However, as our understanding of these models deepens, limitations of traditional CNN architectures become apparentparticularly their difficulty in capturing complex, interconnected patterns within images, which are critical when dealing with the subtle symptoms presented by diseased plant leaves [Chen et al., 2021a].

Several significant challenges persist in applying deep learning to plant disease detection: the scarcity of large, diverse, and well-annotated datasets spanning various plant species and disease types; limited exploration of optimal model configurations tailored for agricultural contexts; and the ongoing struggle to develop systems capable of generalizing effectively under diverse real-world conditions.

This work addresses these critical gaps by systematically exploring the development of optimized deep learning models for plant disease detection. Our investigation focuses on identifying the most effective model architectures and hyperparameters through rigorous evaluation on diverse, publicly available datasets. The primary research objectives include:

- Systematically evaluating various CNN architectures and parameter configurations to determine optimal settings for plant disease classification
- Assessing the impact of different model components on detection accuracy.
- Comparing the performance of advanced deep learning models with established CNN-based approaches.
- Developing practical recommendations for deploying effective plant disease detection systems in real agricultural environments.

Our key contributions advance the state of the art in deep learning for agriculture, demonstrating how carefully designed models can significantly improve the accuracy and reliability of automated disease detection. By achieving consistently high accuracy across multiple datasets, we establish new performance benchmarks.

The potential applications of this research extend well beyond academic exploration. The optimized models developed herein provide practical solutions for precision agriculture, en-abling farmers to make timely, informed decisions. These systems can be integrated with mobile applications, drone-based monitoring platforms, and IoT devices to form comprehen-sive, real-time crop health management frameworks.

This research contributes to a broader vision in which artificial intelligence becomes an indispensable tool in global food security efforts. By demonstrating the practical viability of advanced AI techniques in agriculture, we help pave the way for widespread adoption of intelligent crop monitoring technologies that support sustainable farming and help feed a growing world population.

Organization

This work is organized as follows:

❖ **Chapter 1:** presents an introduction to the research topic, outlining the importance of plant dis-ease detection, challenges in agriculture, and the motivation behind applying ma-chine learning and deep learning techniques.

- ❖ **Chapter 2:** introduces the fundamental concepts of plant diseases and reviews the current state of machine learning and deep learning applications in agricultural disease detection, including detailed discussions of CNN architectures and recent advances.
- ❖ **Chapter 3:** describes the implementation and experimental methodology, including dataset preprocessing, development of custom and pretrained CNN models (DenseNet121, AlexNet, VGG19), performance evaluation, and comparative analyses with tradition-al methods.
- ❖ The work concludes with a summary of findings, practical insights, and recommendations for future research directions in smart agriculture.

Through this work, we aim not only to enhance automated plant disease detection capabilities but also to contribute meaningfully to the global challenge of ensuring food security in the face of climate change and population growth. The development of intelligent, efficient, and reliable crop monitoring systems is imperative, and this thesis represents a step forward in realizing these goals.

Chapter 1

Fundamentals of Plant Disease

1.1 Introduction

Plant diseases are a major threat to global food security and agricultural sustainability, causing significant losses in crop yield and quality each year. Early and accurate identification of these diseases is essential for effective management and minimizing their impact. Traditional methods of disease detection, relying on expert visual inspection and laboratory analysis, are often time-consuming, subjective, and require specialized knowledge. Recent advances in computer vision and deep learning, particularly Convolutional Neural Networks (CNNs), have revolutionized image-based plant disease identification by enabling automated, fast, and precise diagnosis directly from leaf images. This chapter introduces the fundamentals of plant diseases, their symptoms, and current detection techniques, laying the groundwork for understanding how CNN-based image analysis can improve plant disease identification. The objective is to highlight the challenges and opportunities of integrating advanced image processing with plant pathology to enhance agricultural practices.

1.2 Types of Plant Diseases

Plant diseases are commonly caused by fungi, bacteria, viruses, as well as insects and other animals. Even healthy plants can suffer from such issues. Timely intervention can effectively alleviate the damage and restore plant health. This section presents some of the most common plant diseases and offers guidance on how to manage them [Love The Garden, 2025].

1.2.1 Spider Mites

Spider mites (see Figure 1.1), also known as red spider mites, are tiny arachnids measuring between 0.3 and 0.5 mm that infest plants. Their color varies depending on the species and can be yellow-green, brown, or red. Spider mite infestations are characterized by the presence of very fine webs on the plant surface. The longer the mites remain, the thicker and more extensive these webs become. In addition, small brown or yellow spots appear on the upper surfaces of leaves due to the mites piercing the plant tissue and extracting sap. This feeding damage causes leaves to become unsightly and eventually die.

Spider mites thrive in dry, warm environments, which makes houseplants particularly susceptible, although outdoor plants can also be affected during periods of high temperature. These pests often favor old potting soil and can proliferate if plants experience repeated drying out.

Examples of plants susceptible to spider mite infestations include banana plants, hydrangeas, monstera, philodendrons, ficus, orchids, schefflera, palms, dracaena, and ivy. Plants that are frequently dry or under water stress are especially vulnerable [Love The Garden, 2025]

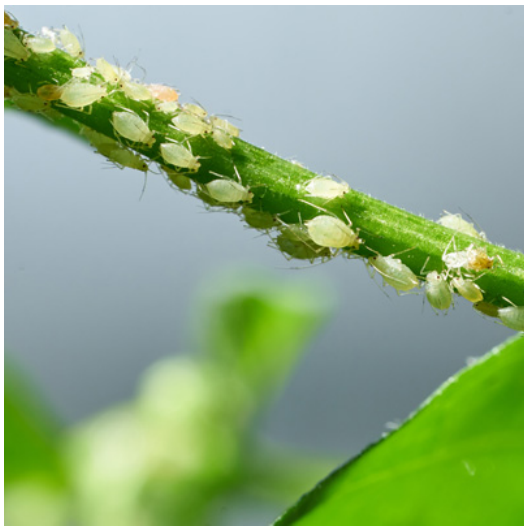


Fig 1.1: Spider Infestation and Its Impact on Plant Health [Love The Garden, 2025].

1.2.2 Rust

Rust is a parasitic fungal disease that develops in warm and humid environments (see Figure 1.2), typically during summer and fall. It is characterized by the appearance of spots or raised areas on the leaves, which vary in color from orange to dark brown. This distinctive coloration gives the disease its name. Over time, from fall to spring, the spots tend to darken. Yellow brown pustules may also form on the underside of the leaves. The fungus can infect not only Chapitre 01 : Fundamentals of Plant Disease the leaves but also the stems. In severe infections, leaf drop occurs, and the plant may ultimately die. Both garden and indoor plants are susceptible to rust infections.

Plants highly vulnerable to rust include roses, snapdragons, hollyhocks, geraniums, carnations, mint, chrysanthemums, irises, hyacinths, dianthus, periwinkles, and plum trees [Love The Garden, 2025].

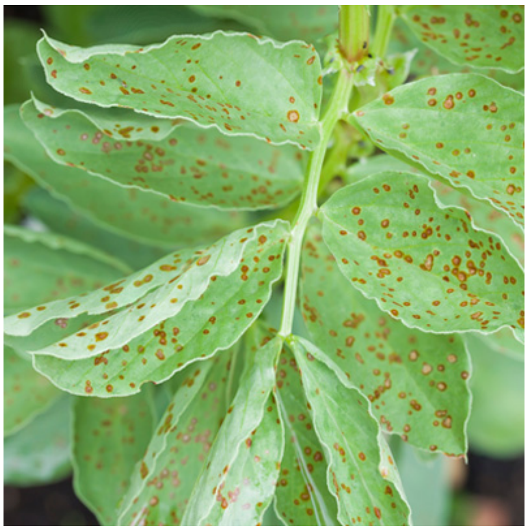


Fig 1.2: Rust Disease, Symptoms and Impact on Plant Health [Love The Garden, 2025].

1.2.3 Powdery Mildew

Powdery mildew is a fungal disease characterized by a white, powdery growth that typically appears on the upper surfaces of young leaves (see Figure 1.3). As the infection progresses, the fungal growth darkens, and the affected leaves begin to curl. Plants that experience morning dampness followed by warm conditions (above 15 °C) during the afternoon, or those that remain wet for extended periods during the day, are particularly susceptible to infestation.

A similar disease, known as downy mildew, differs in that the fungal growth appears on the underside of the leaves, while the upper leaf surfaces develop yellow spots that may later turn brown.

Powdery mildew is a common parasitic fungal disease affecting both vegetable and ornamental plants. In ornamental gardens, plants prone to powdery mildew include roses, lilacs, garden geraniums, asters (especially in the fall), phlox, barberry, bergamot, maple, and honeysuckle.

In vegetable gardens, fruit and vegetable crops frequently affected by powdery mildew include apples, strawberries, grapes, zucchinis, cucumbers, and tomatoes. Indoor plants such as kalanchoe and begonias are also susceptible to this disease [Love The Garden, 2025].

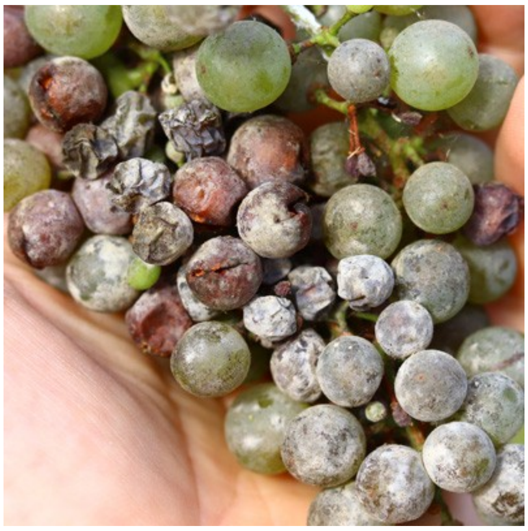


Fig 1.3: Effects of Powdery Mildew on Plant Health [Love The Garden, 2025].

1.2.4 Bacterial Fire Blight

Bacterial Bacterial fire blight is a persistent and serious plant disease caused by the bacterium Erwinia amylovora (see Figure 1.4). Unlike most common plant diseases, which are fungal in origin, this disease is bacterial. This pathogen can cause sudden and widespread outbreaks, affecting fruit and tree crops as well as gardens, public green spaces, and landscapes, thereby posing a threat to all susceptible plants in an area.

The bacterium is transmitted by insects and birds, as well as by environmental factors such as wind and rain. The disease is named for its characteristic symptoms, which resemble plants being scorched by fire. Typical signs include brown-black discoloration, wilting, and shriveling of flowers, leaves, branches, and fruits. In the spring, affected branches may initially exhibit watery, sticky tissues. Additionally, white to yellowish bacterial ooze droplets may appear on infected plant parts.

Plants susceptible to infection, referred to as host plants, also contribute to the spread of the disease. All known host plants of bacterial fire blight belong to the Rosaceae family (rose family).

Pears (Pyrus spp.) are the most susceptible to this disease. Other vulnerable shrubs and trees within the Rosaceae family include Cotoneaster (dwarf cotoneaster), Pyracantha (firethorn), Chaenomeles (Japanese quince), Photinia (Indian hawthorn), Crataegus (hawthorn), Sorbus (mountain ash), Amelanchier, Malus (apple and ornamental apple), and Cydonia (quince) [Love The Garden, 2025].

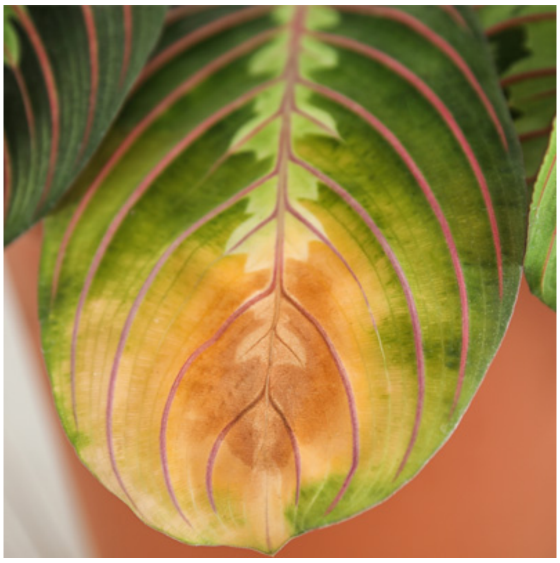


Fig 1.4: Impact of Bacterial Fire Blight on Plants.

1.2.5 Marsonia (Black Spot Disease)

Marsonia, also known as black spot disease, and sooty mold (see Figure 1.5) are fungal diseases that cause brown to black, star-shaped spots on the leaves, branches, or stems of affected plants. This disease is particularly common in roses. The characteristic black spots give the disease its name, "*blackspot*". Infected leaves typically turn yellow and eventually fall off. The fungi thrive on honeydew secreted by sap-sucking insects such as aphids, whiteflies, scale insects, psyllids, and leafhoppers. Effective control of sooty mold requires managing these insect populations, which are the primary source of honeydew.

Besides roses, sooty mold can affect many plants with firm leaves, including houseplants, heather, ornamental container plants (such as citrus), and Mediterranean plants like olive trees and oleanders. In vegetable gardens, tomatoes are particularly susceptible to sooty mold infections [Love The Garden, 2024].

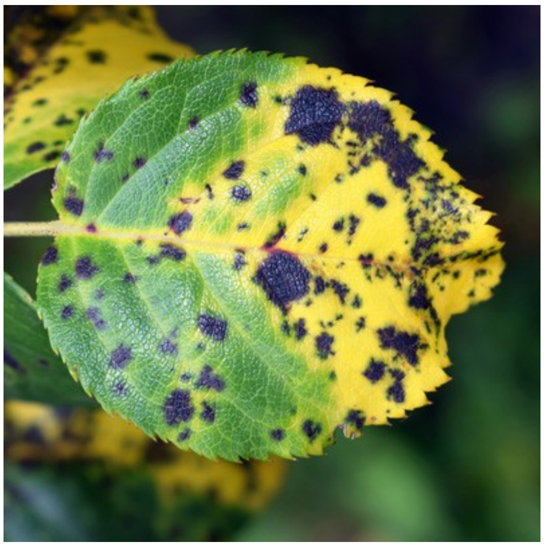


Fig 1.5: Impact of Marsonia on Plant Health [Love The Garden, 2025].

1.2.6 Botrytis (Gray Mold)

Botrytis cinerea is a typical opportunistic fungal parasite, recognizable by its characteristic gray, fuzzy growth. This fungus causes affected plant parts to rot (see Figure 1.6). In addition to the gray, fuzzy mycelium, botrytis infection results in severe leaf discoloration, browning of fruits, and white spots on petals. The disease is also known by synonyms such as gray rot, gray mold, and noble rot.

This fungal disease often develops during warm, humid weather and commonly spreads on senescing or dying flowers. The fungus requires stagnant moisture to thrive. To prevent infection, it is advisable to water potted plants from the bottom and maintain adequate spacing between plants to ensure good air circulation. Increasing ventilation is particularly important for plants grown under glass or in greenhouses. When removing infected plant material, exercise caution by placing the debris directly into a sealed plastic bag to prevent further spread.

For control, spraying healthy plant parts with neem oil-a natural pesticide effective against fungi, aphids, mites, thrips, and other pests can be beneficial. Additionally, products like Substral Naturen Cuprex Garden may be used to combat botrytis.

Potted plants and bedding plants are especially susceptible to this fungal disease [Love The Garden, 2024].

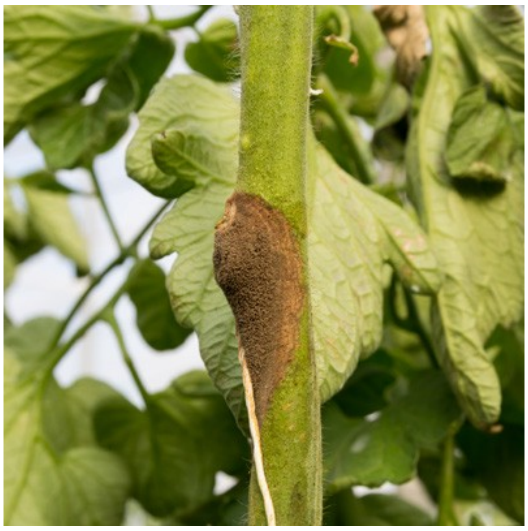


Fig 1.6: Impact of Botrytis on Plant Health [Love The Garden, 2025].

1.2.7 Scab

Scab is a fungal plant disease that primarily occurs during periods of leaf wetness (see Figure 1.7). The fungi responsible for scab thrive in humid conditions. Infection typically begins at the edges of leaves and then spreads toward the center. Initially, olive-green spots appear, which later turn brown or black. Infected leaves wilt and curl, even under humid conditions. Subsequently, branches and fruits become affected, developing spots as well. The plant stems may also discolor and, in severe cases, may die back or drop off.

White mold is often observed on the leaf edges in association with the disease. On apples and pears, scab causes characteristic cracks and corky spots on the fruit surface. These infections lead to premature leaf drop and weaken the overall health of the tree.

Scab can also affect potatoes and tomatoes. Tomatoes develop brown spots, shrink, and eventually rot, while potatoes form brown lesions that progress into slimy, foul-smelling rot.

This disease is commonly found in fruit crops, vegetables, and ornamental plants. To control scab, products such as Substral Naturen Cuprex Garden can be applied in vegetable gardens or on fruit trees to combat the fungal infection [Love The Garden, 2024].

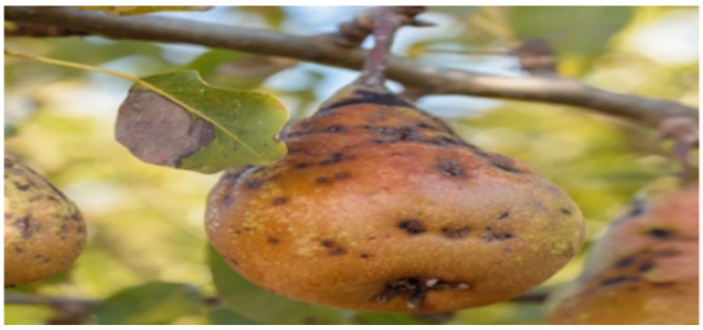


Fig 1.7: Impact of Scab on Plant Health [Love The Garden, 2025].

1.2.8 Iron Deficiency

Young leaves of acid-loving plants (see Figure 1.8), such as azaleas, rhododendrons, and camellias, often exhibit chlorosis-yellowing of the leaf tissue-while the veins remain green, which is a classic symptom of iron deficiency. Iron is essential for the synthesis of chlorophyll, the pigment critical for photosynthesis. If the deficiency persists, it can lead to progressive yellowing of mature leaves, stunted plant growth, and eventual leaf death.

Iron deficiency is most commonly observed in soils that are rich in lime, have a high PH, or are excessively waterlogged, conditions that reduce oxygen availability and limit iron uptake. Acid-loving plants, including rhododendrons, azaleas, hydrangeas, skimmias, and kalmias, as well as certain fruit trees like apple and pear, are particularly vulnerable when grown in alkaline soils (pH above 7). In rhododendrons, chlorosis may occur even when soil pH exceeds 5.5.

These observations underscore the importance of proper soil management to prevent nutritional imbalances that can adversely affect the growth and health of acid-loving plants.

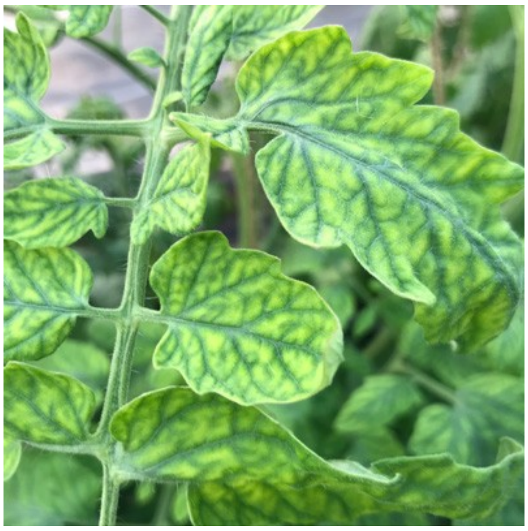


Fig 1.8: Impact of Iron Deficiency on Plant Health [Love The Garden, 2025].

1.3 Symptoms and Indicators of Plant Diseases

Symptoms refer to the visible changes in a plant's appearance or behavior resulting from the presence of a pathogen or environmental stressor. Common symptoms include abnormal leaf growth, discoloration, wilting, and other signs indicating potential plant health issues [Reis-Pereira et al., 2024].

Indicators, on the other hand, are specific signs or patterns that suggest the presence of a particular disease or pathogen. These include visible structures of the pathogen, such as fungal spores, bacterial ooze, or viral streaks on plant tissues.

Effective diagnosis and management of plant diseases require careful examination of the affected plants. Observing the undersides of leaves and documenting visible symptoms or signs can provide valuable clues for identifying the underlying problem.

1.3.1 Spider Mites Treatment

Immediately remove affected leaves and replace old soil. Maintain a humid environment by misting plants at least once a week, preferably using rainwater. Relocate plants to a cooler area $(around12 - 14\hat{A}\check{r}C)$, as spider mites dislike humidity and cooler temperatures. Isolate affected plants promptly, since mites spread rapidly. Use a biological insecticide designed for mites, such as Substral Naturen Polysect Organic Insecticide Spray. For prevention, pre-treat plants with Substral Naturen nettle manure.

1.3.2 Rust Treatment

Control rust through a combination of cultural practices, natural remedies, and chemical treatments. Remove infected leaves and improve air circulation to reduce humidity. Water plants at the base to keep foliage dry. Natural treatments include spraying a baking soda and water mixture or neem oil. Garlic extract and sulfur-based fungicides are also effective. Employ biological controls like beneficial microbes and choose rust-resistant plant varieties. Crop rotation and mulching can further reduce fungal spread.

1.3.3 Powdery Mildew Treatment

Remove and dispose of infected leaves to minimize fungal spread. Improve air circulation by spacing and pruning plants. Avoid overhead watering; water at the base instead. Natural remedies such as neem oil, baking soda solution (1 tbsp baking soda + 1 tsp soap in 1 gallon water), or milk solution (1 part milk to 9 parts water) can be applied.

Sulfur-based fungicides and commercial treatments like potassium bicarbonate also help control the disease. Use resistant varieties and practice crop rotation to reduce future outbreaks.

1.3.4 Bacterial Fire Blight Treatment

Remove affected plant parts promptly to prevent spread; severe infections may require removal of the entire plant. Sanitize pruning tools between cuts to avoid bacterial transmission. Dispose of infected material properly. Copper based bactericides can reduce infection risk. Avoid overhead irrigation to limit bacterial dispersal.

1.3.5 Marsonia (Black Spot Disease) Treatment

- **Remove infected leaves and branches**, ensuring debris is discarded and not composted.
- **Improve air circulation** by spacing plants and pruning excess growth.
- **Apply fungicides** effective against leaf spot diseases, such as copper, chlorothalonil, or mancozeb, following label instructions.
- **Avoid overhead watering**; water at the plant base to keep foliage dry.
- **Implement crop rotation** and regularly remove fallen leaves to reduce fungal buildup.

Combining these methods helps control Marsonia and protect plants from further damage.

1.3.6 Botrytis (Gray Mold) Treatment

- **Remove infected tissues**, discarding affected flowers, leaves, and stems; sanitize tools after pruning.
- **Enhance air circulation** by spacing plants and pruning dense foliage.
- **Water at the base**, avoiding overhead irrigation; water early in the day for quick drying.
- **Apply fungicides** targeting Botrytis, such as boscalid, cyprodinil, or chlorothalonil, following manufacturer guidance.

- **Maintain sanitation** by removing fallen debris.
- **Use protective fungicides preventively** during humid, wet conditions.

Proper care, pruning, and fungicide use reduce Botrytis risk effectively.

1.3.7 Scab Treatment

- **Remove and dispose of infected material**, especially fallen leaves and fruit; avoid composting infected debris.
- **Apply fungicides** formulated for apple scab, including captan, myclobutanil, or tebuconazole, early in the season and after rainfall.
- **Prune and thin trees** to improve airflow and light penetration.
- **Avoid overhead watering**; water at the base early in the day.
- **Plant resistant varieties** when possible.
- **Sanitize tools and equipment** regularly to prevent disease spread.

These steps help control apple scab and protect fruit trees.

1.3.8 Iron Deficiency Treatment

Treat iron deficiency by applying iron supplements such as chelated iron or iron sulfate to soil or as foliar sprays, following product directions. If soil pH is high, reduce it using sulfur or acidifying fertilizers to improve iron availability. Ensure proper soil drainage and aeration to facilitate iron uptake. Adding organic matter like compost improves soil structure and nutrient availability. Avoid excessive nitrogen or phosphorus fertilization, which can inhibit iron absorption. Maintain balanced watering to support nutrient uptake without causing waterlogging.

1.4 Role of Technology in Plant Disease

Increasing agricultural productivity while preserving natural resources and improving the quality of life for farmers remains a major challenge for sustainable agriculture. Plant diseases significantly impact crop yields and continue to be a persistent concern for farmers. Optimizing crop management, particularly minimizing the use of fertilizers and pesticides, is a critical focus in modern agriculture.

Improved early detection of plant diseases represents a significant research area aimed at enhancing treatment effectiveness. For example, tomatoes and potatoes are susceptible to blight, and potatoes can be affected by various diseases that lead to tuber rot. Tools and techniques for detecting the phytopathogenic microorganisms responsible for these diseases are already deployed in fields, while others are still under development [Conversation, 2020].

1.4.1 Biosensors in Plant Disease Detection

Among currently used tools, biosensors convert biological or physical elements such as antibodies, proteins, or DNA into measurable signals. Biosensors that detect proteins (antibodies) or nucleic acids (DNA or RNA) primarily rely on ELISA (Enzyme-Linked Immunosorbent Assay) and PCR (Polymerase Chain Reaction) techniques. ELISA detects phytopathogens in plant extracts using antibodies specific to the pathogen, while PCR amplifies targeted regions of the pathogen's genome from DNA extracted from diseased plants, facilitating detection and quantification.

However, these diagnostic tools require prior knowledge of the pathogen to develop the necessary antibodies or PCR primers. They are also destructive, involving grinding of plant tissues to release the pathogen or its nucleic acids, and demand expensive laboratory equipment and skilled labor.

Although these analyses are generally performed in laboratories, portable field detection kits using biological markers have been developed. These kits can detect pathogens such as Phytophthora species, fire blight caused by Erwinia amylovora, and Potato Y virus on-site. These methods are faster, more cost-effective, and less labor-intensive than laboratory procedures. Future advances may enable fine-level pathogen identification through smartphone images uploaded to centralized databases, although such systems are not yet widely implemented in plant disease diagnostics [6].

1.4.2 Sensing Plant Diseases with Technology

A promising technology under development is gas chromatography (GC), which analyzes gases or gas mixtures to differentiate among them. Some plant diseases emit volatile organic compounds (VOCs) that can be detected by GC. For instance, potato tuber rot caused by bacterial pathogens such as Ralstonia solanacearum and Clavibacter michiganensis releases specific VOCs before visible symptoms develop. These compounds can be detected by electronic noses, allowing early pathogen detection prior to symptom appearance.

Although still in the testing phase, electronic nose technology is already employed in potato storage facilities. The next step involves deploying electronic noses directly in soil to detect diseases in the field.

1.4.3 Visualizing Plant Diseases

Spectro-imaging is another advanced technology for plant disease detection, capable of identifying infections before visible symptoms appear. Hyperspectral imaging is the most precise among these methods, capturing images across a broad range of spectral bands, including those beyond the visible spectrum such as near-infrared

Stressed plants emit distinct biochemical and structural radiation patterns compared to healthy plants, which can be identified through hyperspectral imaging. This spectral information enables disease identification based solely on imaging data, marking a major advancement in plant pathology.

Hyperspectral systems, primarily composed of specialized cameras, can be mounted on mobile platforms such as drones to analyze entire fields nondestructively. This approach offers significant advantages over traditional random sampling methods.

Recent studies have shown that hyperspectral imaging can detect potato blight and Alternaria infections before symptom onset. Chinese researchers developed a hyperspectral method to differentiate between two visually similar tomato diseases blight and Alternaria-based on their distinct spectral signatures. In the Netherlands, hyperspectral cameras mounted on tractors are used to detect plants infected with Potato Y virus.

These technological advances facilitate faster disease detection in the field, enabling early interventions that help limit disease spread [Conversation, 2020].

1.5 Challenges in Identifying Plant Diseases

Accurate identification of plant diseases is a fundamental component of effective agricultural management. Traditionally, disease identification has been performed manually through visual inspection or microscopy.

1.5.1 Extrinsic Factors

1.5.1.1 Image Background

In most image-based leaf analysis methods, the initial step is leaf segmentation. When images are captured against a plain background such as a white or blue panel segmentation can typically be performed automatically with minimal difficulty. However, when the background includes complex elements such as other plants, soil, or overlapping leaves, segmentation becomes significantly more challenging [Barbedo, 2016].

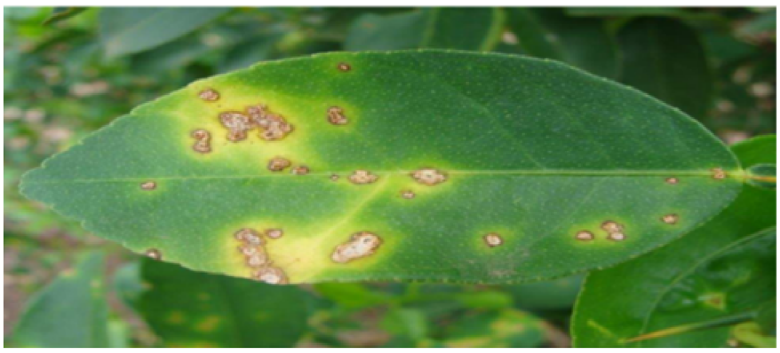


Fig 1.9: Image background analysis for plant disease detection.

1.5.1.2 Image Capture Conditions

Various factors influence image characteristics, making accurate analysis more difficult. Ideally, all images should be captured under uniform conditions, which is feasible in controlled environments like laboratories. However, real-world conditions vary widely, especially in outdoor settings. Understanding the primary factors affecting segmentation and developing methods to address them is essential for improving accuracy in practical applications [Barbedo, 2016].

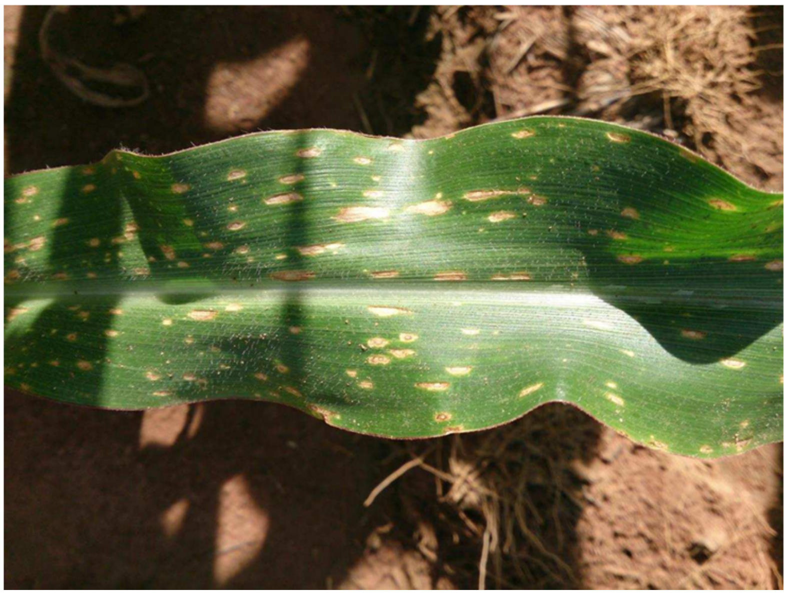


Fig 1.10: Exemple of a leaf image with spcular reflections and sevral light/shadow transitions.

1.5.2 Intrinsic Factors

1.5.2.1 Symptom Segmentation

Many plant disease symptoms do not have clearly defined edges and instead blend gradually into healthy tissue [Streets, 1972]. This lack of distinct boundaries makes precise segmentation difficult. Even when manually outlined, there is often ambiguity in defining symptom borders. Although automated methods may offer consistency, variations in symptoms introduce additional complexity, affecting the accuracy of thresholding and other image-processing techniques.

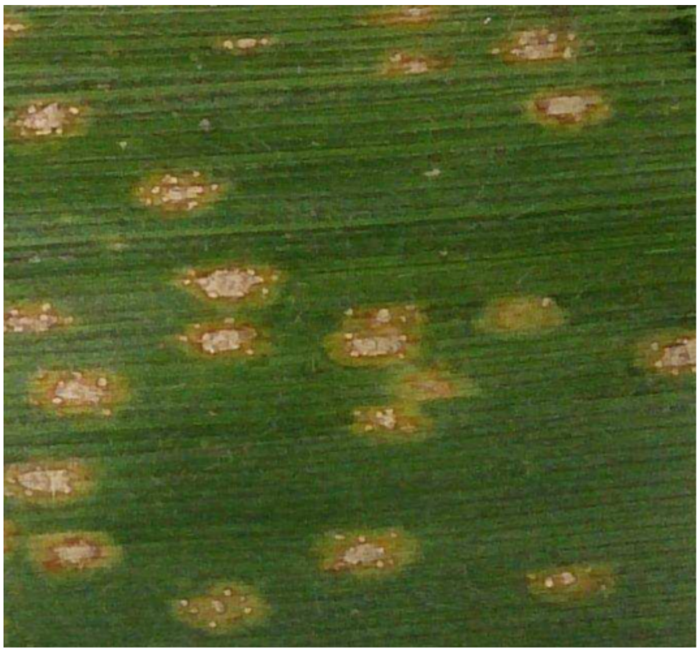


Fig 1.11: Exemple of symptoms with no clear edges.

1.5.2.2 Symptom Variations

While some plant diseases display distinct and recognizable symptoms, considerable variation often occurs in color, shape, and size. These variations result from factors such as plant species, environmental conditions, and the interactions between the plant and pathogen. Changes in any of these factors can alter symptom expression, making accurate identification more challenging when relying solely on visible-spectrum imaging.

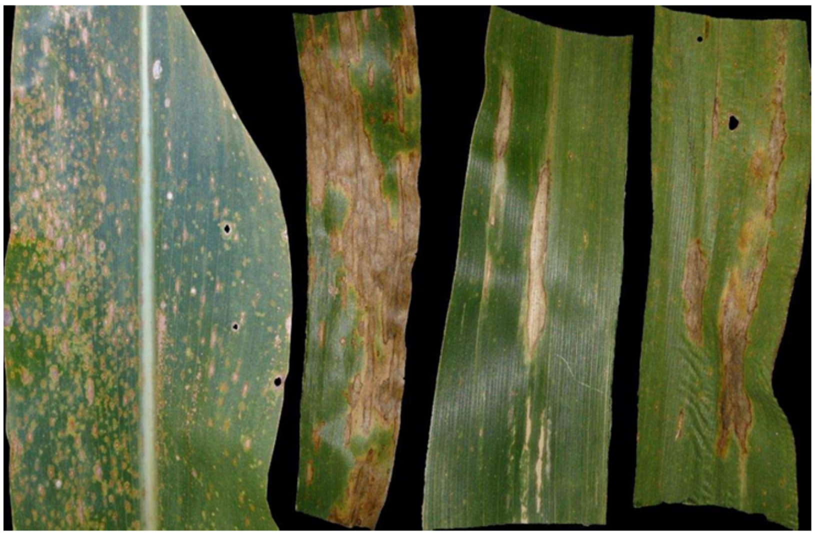


Fig 1.12: Variation in symptoms of Southern corn leal blight.

1.5.2.3 Multiple Simultaneous Disorders

It is common for multiple disorders to affect the same plant simultaneously. The presence of more than one disease, combined with other issues such as nutritional deficiencies and pest infestations, complicates accurate identification. When a plant's immune system is weakened by one infection, it becomes more susceptible to additional problems. Observations indicate that symptoms caused by different diseases often coexist, making it difficult to isolate the cause of specific visual patterns [Chester, 1933].

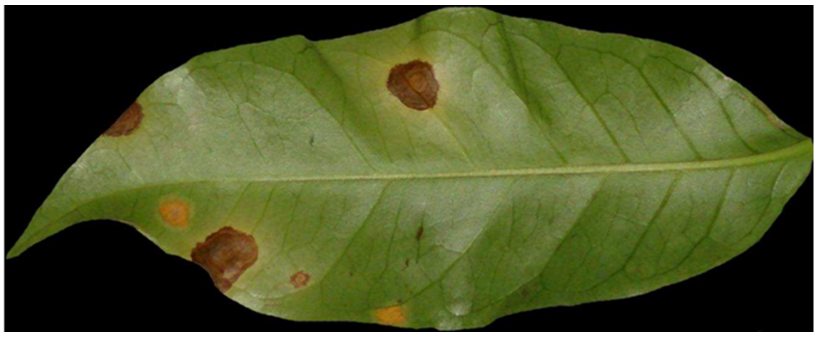


Fig 1.13: Coffee leaf containing symptoms of rust and Cercospora leaf spot.

1.5.2.4 Different Disorders with Similar Symptoms

Many plant disorders including diseases, nutritional deficiencies, pest infestations, and environmental stresses produce similar symptoms. These overlaps make it challenging to accurately determine the exact cause of a symptom based solely on visible characteristics. The wide variety of potential conditions increases the complexity of diagnosis, particularly when relying only on standard imaging techniques [Barbedo, 2016][Dordas, 2008].

1.6 Conclusion

This chapter has underscored the critical importance of accurate plant disease identification and the inherent challenges posed by symptom variability, multiple cooccurring disorders, and visually similar symptoms across different diseases. While conventional diagnostic methods remain foundational, they are often limited by subjectivity and practicality. The emergence of image-based technologies, especially those utilizing Convolutional Neural Networks, offers a powerful solution to overcome these limitations by enabling rapid, objective, and scalable disease detection from plant images. However, successful implementation requires addressing challenges such as diverse environmental conditions, complex backgrounds, and symptom variations. The integration of CNNs into plant disease diagnostics promises significant advancements in precision agriculture, supporting timely interventions and ultimately contributing to improved crop health and food security.

This rising importance of automation points toward a shift in how plant health is monitored and maintained. As we move forward, Chapter 2 will delve into the application of machine learning techniques in plant disease detection. It will discuss how these advanced methods are shaping the future of precision agriculture and plant pathology.

Chapter 2

Machine Learning in Plant Disease Detection

2.1 Introduction

In recent years, the integration of deep learning techniques, particularly Convolutional Neural Networks (CNNs), has revolutionized the field of plant disease detection. Accurate and timely identification of plant diseases is critical to ensuring agricultural productivity, food security, and the economic well-being of farmers worldwide. Traditional manual inspection methods are labor-intensive, time-consuming, and prone to error, especially over large agricultural areas. CNNs, with their powerful ability to automatically extract hierarchical features from image data, offer a promising solution by enabling precise, scalable, and real-time disease diagnosis. This chapter explores the fundamental principles of deep learning, delves into the architecture and advantages of CNNs, and reviews the latest advancements in applying these techniques to plant disease detection.

2.2 Overview of Machine Learning in Agriculture

Agriculture plays a critical role in human survival by providing food, clothing, and shelter. However, crop yields are influenced by numerous abiotic and biotic factors, including soil conditions, climate variability, pests, and diseases. To safeguard agricultural productivity, modern technologies such as the Internet of Things (IoT) and machine learning (ML) are increasingly employed for site monitoring, pest identification, and damage prediction [Yao et al., 2023].

Machine learning, a subfield of artificial intelligence (AI), has found broad applications in agriculture, including harvesting automation, yield estimation, pest and weed control, irrigation optimization, and plant disease detection. Among various plant organs, leaves are the most commonly analyzed for disease diagnosis due to their essential role in photosynthesis and overall plant growth. Leaf diseases can severely impact plant health and productivity, making early detection crucial.

Traditionally, plant disease identification has relied on manual inspection by experts. However, this approach is often labor-intensive, time-consuming, costly, and prone to inaccuracies-especially in areas where human presence is limited due to environmental or health constraints [Salman et al., 2023]. Machine learning offers a promising alternative by enabling efficient, scalable, and accurate disease recognition.

2.3 Importance of Automation in Plant Disease Detection

Monitoring and managing crop health is essential to ensuring global food security, especially given the rapidly increasing world population. In developed countries, technological advancements have been widely adopted in agriculture to support crop detection, monitoring, and management. However, developing nations such as India still face significant challenges in cultivating crops without excessive reliance on chemical fertilizers and pesticides.

Large-scale farming is inherently labor-intensive, and many farmers rely on minimal labor due to financial or logistical constraints. Visual inspection of crops across vast fields is often impractical, leading to undetected infections and substantial yield losses. This underlines the pressing need for automation in agriculture particularly in the context of crop monitoring and the detection of plant diseases and nutrient deficiencies.

Plant diseases can disrupt normal physiological processes in crops and may manifest on various plant parts, including leaves, stems, roots, flowers, and fruits. Disease outbreaks can quickly spread and destroy large crop areas. According to estimates, plant diseases contribute to a $20\% - 40\%$ reduction in global agricultural yields, causing considerable economic losses. For example, brown spot disease in rice can reduce yields by up to 50%, while grapevine diseases often lead to extensive damage in vineyards. In addition to lowering yield quantity, diseases may also affect the quality and marketability of agricultural products [Yağ and Altan, 2022].

Traditionally, farmers have relied on visual inspections to detect signs of disease. However, this method is inefficient and error prone especially for large farms due to human limitations, variable expertise, and the demand for continuous monitoring. Automated systems, supported by image processing and sensing technologies, offer an effective alternative by enabling real-time disease detection and accurate data logging [Alzoubi et al., 2023].

Barbedo (2013) highlights that image processing not only facilitates the identification of plant diseases but also helps quantify their severity, thus improving disease management strategies [Arnal Barbedo, 2013]. In addition, Kamilaris and Prenafeta Boldu(2018) emphasize the transformative potential of deep learning, IoT, and smart sensor networks in scaling plant health monitoring systems and enhancing their precision [Kamilaris and Prenafeta-Boldú, 2018].

2.4 Fundamentals of Machine Learning and Deep Learning

Agriculture is a vital sector contributing significantly to global economies and feeding billions of people worldwide. However, it faces numerous challenges such as population growth, excessive pesticide use, climate change, and plant diseases. With the rising demand for agricultural products, monitoring plant health has become a fundamental necessity, with leaf disease detection playing a major role. Due to climatic changes and various diseases, different types of infections affect crop leaves, adversely impacting food production. Therefore, there is an urgent need for innovative ideas and approaches to address these challenges.

Plant diseases are caused by biotic agents including bacteria, fungi, nematodes, and viruses, as well as abiotic factors or plant stresses such as allelochemicals, climatic conditions, and agricultural practices [Salman et al., 2023]. Early disease detection is critical, not only to preserve crop yield but also to facilitate mitigation strategies.

Deep learning, a subset of machine learning based on artificial neural networks, attempts to mimic the human brain's ability to learn from data. Among deep learning techniques, convolutional neural networks (CNNs) have demonstrated promising results in plant pathology tasks such as object detection and image classification. Before CNNs, various image processing based automatic methods were developed for detecting and classifying plant diseases [Shoaib et al., 2023].

2.4.1 Definition and Key Concepts

Given the detrimental effects of plant diseases on crop yield, food security, and farmers' livelihoods, developing automatic plant disease recognition systems is essential, especially where resources for manual disease detection are limited. Recently, deep learning based approaches that perform multi-class classification, single-dimensional localization, segmentation, and pixel-wise classification for early disease detection have been explored, often leveraging transfer learning techniques [Salman et al., 2023]. These methods aim to control plant disease damage and contribute to food safety and farmer economics.

Furthermore, smart mobile applications for multi-class disease detection in crops such as tomatoes and grapes have been developed. These systems improve reliability, support both user and camera levels, consider practical challenges overlooked by existing approaches, and are often open source. However, current prediction services suffer from low accuracy, limited class detection, and slow inference times. Analyzing previous image classification trends has motivated the design of multi-class classification models powered by mobile edge computing platforms to overcome these limitations.

In the machine learning research domain, continuous technological advances have enhanced image processing and automated detection technologies. Consequently, researchers have increasingly applied machine learning techniques to detect diseases in plant branches and foliage effectively. Such plant disease detection systems process images of leaves by capturing, filtering, and classifying infected versus healthy leaves. The trained models then predict or classify new leaf images accurately. Developing these systems requires comprehensive datasets containing images of both infected and healthy leaves for effective training [Li et al., 2021].

2.4.2 Supervised vs. Unsupervised Learning

Machine learning (ML) gained public recognition nearly three decades ago and has since become integral to various aspects of daily life. Over time, different branches of ML have emerged to address diverse problems. The two most prominent types today are supervised and unsupervised learning, which differ primarily in how they learn from data.

In supervised learning, algorithms are trained on labeled datasets where the correct output is known for each input. This enables models to learn patterns, such as distinguishing diseased leaves from healthy ones in images. Two key supervised learning techniques used in plant disease detection are:

1. **Support Vector Machines (SVM):** These algorithms find the optimal boundary separating different classes in the data but are limited when handling complex features.

2. **Deep Learning:** This approach uses layered neural networks capable of processing large volumes of image data with high accuracy, although it demands significant computational resources.

Conversely, unsupervised learning deals with unlabeled data and aims to discover hidden patterns or groupings without predefined categories. In agriculture, unsupervised methods are useful for tasks such as soil analysis, crop identification, and processing satellite imagery, where data clustering or classification occurs without prior labeling [Li et al., 2021].

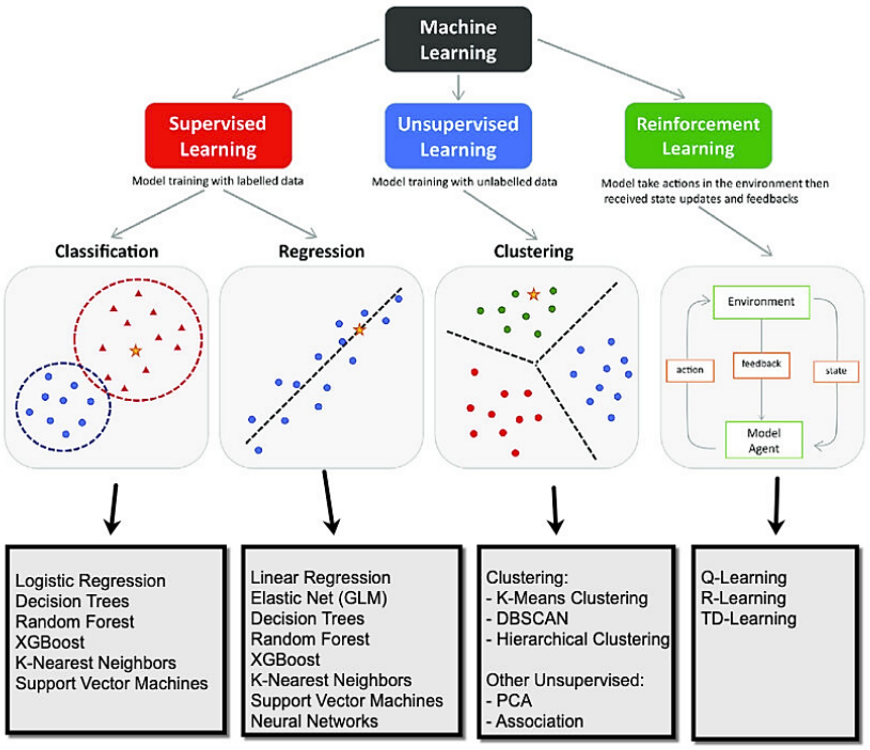


Fig 2.1: Overview of Key Machine Learning Types and Approaches [Peng et al., 2021].

There are three fundamental types of machine learning: **Supervised Learning**, **Unsupervised Learning**, and **Reinforcement Learning**.

- **Supervised Learning:** This type of learning maps a set of inputs (features) to an output (target). It includes two main subtypes: classification and regression.
  - **Classification:** Involves identifying the category to which an observation belongs. A common example is binary classification, which estimates the probability that an observation belongs to one of two classes such as predicting whether a customer is a buyer or non-buyer (class 0 or 1).
  - **Regression:** Focuses on predicting a continuous numeric value. A practical application is forecasting future sales demand, which is a specialized form of regression.
- **Unsupervised Learning:** Involves extracting patterns from unlabeled data. Two widely used techniques are clustering and dimensionality reduction.
  - **Clustering:** Algorithms like K-means are often applied to group similar customers based on shared attributes.
  - **Dimensionality Reduction:** Techniques such as Principal Component Analysis (PCA) are used to reduce the number of input features, which helps im-

prove the efficiency of other machine learning algorithms and facilitates the visualization of clusters.

- **Reinforcement Learning:** This approach enables software agents to learn optimal actions by interacting with an environment and receiving feedback in the form of rewards. It forms the basis of many artificial intelligence systems, where the software continuously learns and improves decision-making through experience.

In the figure, training data is represented by colored dots and triangles, whereas yellow stars symbolize new data points that the trained model can predict [Peng et al., 2021].

2.4.3 Introduction to Deep Learning

Deep learning is a subfield of machine learning that uses algorithms inspired by the structure and function of the human brain, known as artificial neural networks. These models automatically learn to extract features and patterns from large amounts of data through multiple layers of processing. Unlike traditional machine learning, which often relies on manual feature engineering, deep learning networks learn features directly from raw inputs such as images, audio, or text.

Deep learning has gained significant attention in recent years due to its success in tasks such as image and speech recognition, natural language processing, and autonomous driving. This progress has been facilitated by the availability of big data, increased computational power (notably GPUs), and advancements in neural network architectures such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).

At its core, a deep learning model consists of layers of interconnected neurons: an input layer, one or more hidden layers, and an output layer. During training, these models adjust their internal parameters through a process called backpropagation, optimizing them to minimize the difference between predicted and actual outputs [Goodfellow et al., 2016].

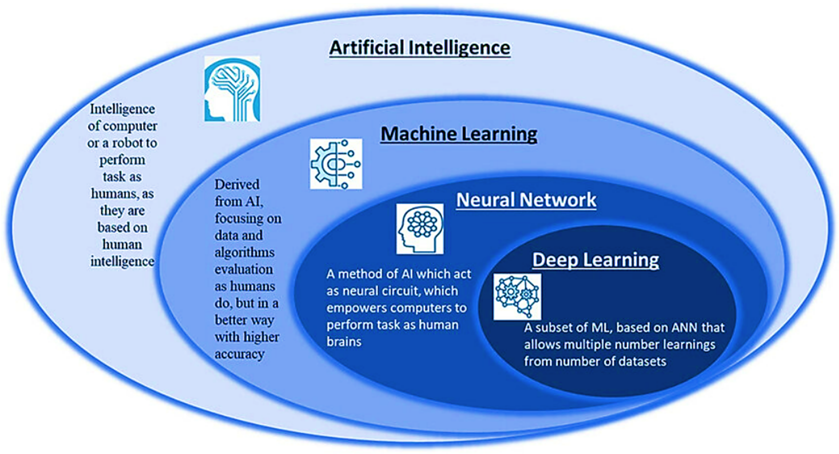

Artificial Intelligence
Intelligence of computer or a robot to perform task as humans, as they are based on human intelligence
Machine Learning
Derived from AI, focusing on data and algorithms evaluation as humans do, but in a better way with higher accuracy
Neural Network
A method of AI which act as neural circuit, which empowers computers to perform task as human brains
Deep Learning
A subset of ML, based on ANN that allows multiple number learnings from number of datasets

Fig 2.2: Artificial Intelligence (AI) vs Machine Learning (ML) vs Neural Network vs Deep Learning [Shang et al., 2024].

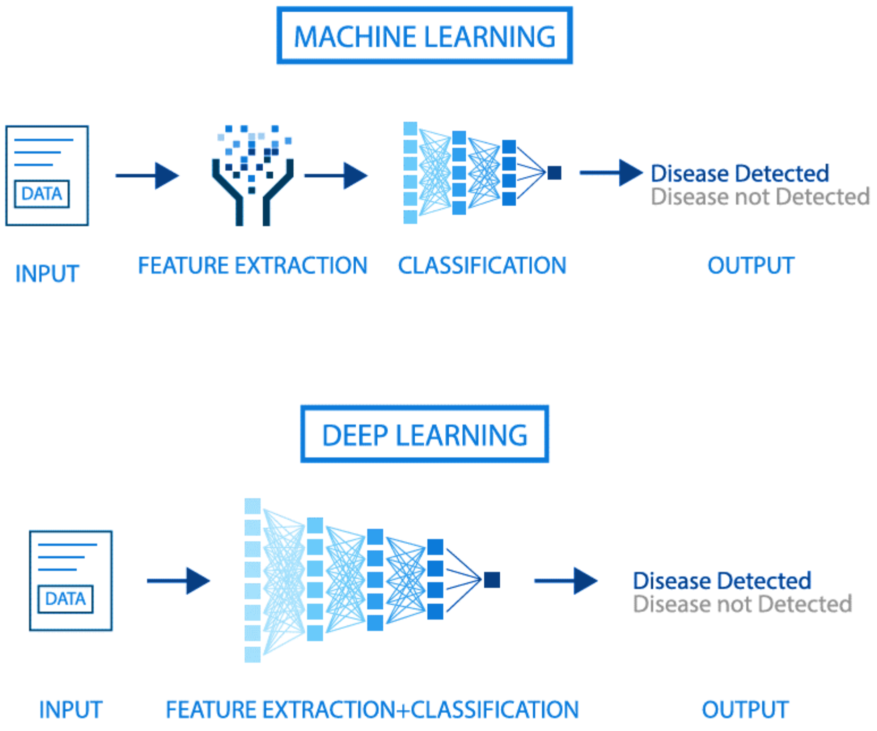


Fig 2.3: Distinguishing Machine Learning from Deep Learning [Khan et al., 2021].

2.5 Convolutional Neural Networks (CNNs) for Image Classification

Convolutional Neural Networks (CNNs) are a specialized subset of artificial intelligence primarily designed for image analysis. They detect patterns by breaking down an image into smaller components, enabling the identification of elements such as edges, textures, and shapes. CNNs have been successfully applied across diverse domains including medical image analysis, autonomous vehicles, and facial recognition [Rizvi, 2020].

2.5.1 Basics of CNNs

CNNs were originally developed for image classification and quickly demonstrated their effectiveness, subsequently expanding to other computer vision tasks such as object detection, image segmentation, and caption generation. A typical CNN architecture consists of convolutional layers, pooling layers, and fully connected layers, with nonlinear activation functions applied after certain operations.

- **Convolutional layers** extract feature maps by applying weighted kernels (filters) combined with activation functions to detect local patterns.
- **Pooling layers** reduce the spatial dimensions of feature maps, commonly using max pooling or average pooling, which helps decrease computational load and improves model robustness.

To further enhance robustness and mitigate overfitting, dropout layers are used, which randomly deactivate a subset of neurons during training.

Despite their strengths, CNNs encounter challenges when dealing with structured noise such as occlusions and partially visible features. These issues often require hand-crafted solutions that may not generalize well to complex or dynamic scenes. Recent advances in light field cameras, which capture four-dimensional data by recording multiple perspective views, improve handling of occlusions and enhance object detection accuracy. However, such cameras can be expensive or impractical for many applications.

As a more accessible alternative, RGB images captured from slightly different lateral positions can approximate some benefits of light field data, preserving structural and complementary visual information that enhances model performance [LeCun et al., 2015].

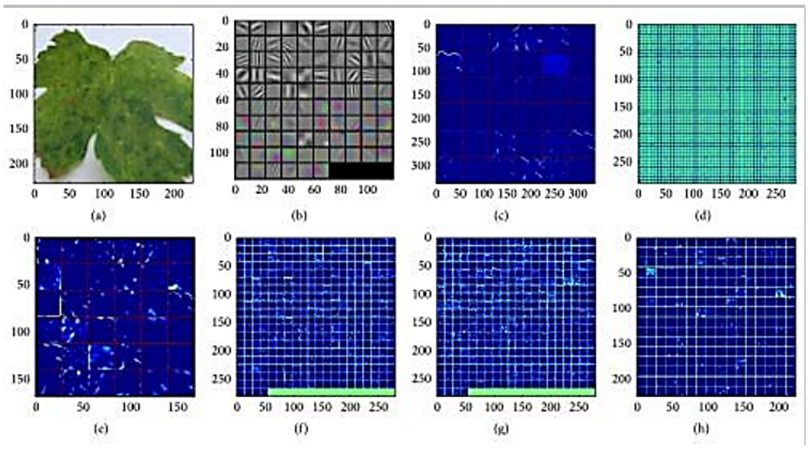


Fig 2.4: Visualization of features in trained classification model: (a) original image; (b) the first layer filters, Conv1; (c) the first layer output, Conv1 rectified responses of the filters, first 36 only; (d) the second layer filters, Conv2; (e) the second layer output, Conv2 (rectified, only the first 36 of 256 channels); (f) the third layer output, Conv3 (rectified, all 384 channels); (g) the fourth layer output, Conv4 (rectified, all 384 channels); (h) the fifth layer output, Conv5 (rectified, all 256 channels) [Sladojevic et al., 2016].

2.5.2 Architectural Framework of CNNs

Convolutional Neural Networks (CNNs) are widely used in image classification tasks and have demonstrated high effectiveness across numerous research studies due to their powerful feature extraction capabilities [LeCun et al., 1988].

The architecture of a CNN (illustrated in Figures 2.5 and 2.6) typically consists of convolutional layers, pooling layers, and fully connected artificial neural network layers [Masko and Hensman, 2015]. The most crucial component is the convolutional layer, which gives the network its name. This layer is responsible for extracting hierarchical features from input images, forming the foundation of the CNN's ability to learn visual patterns.

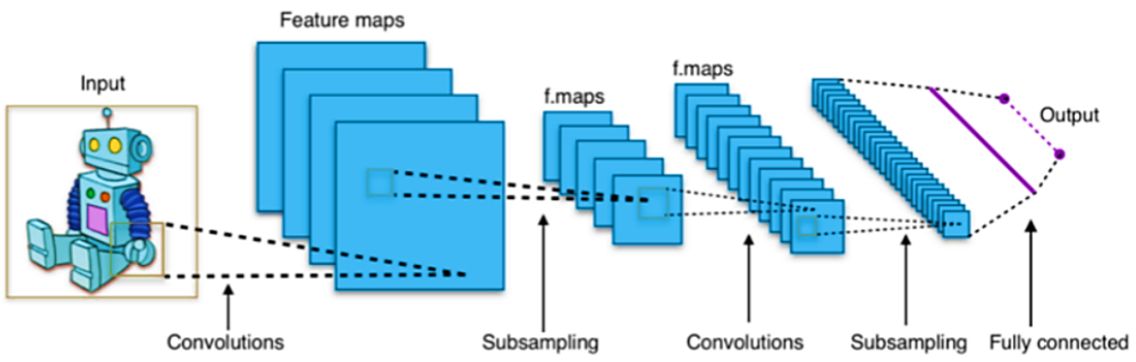


Fig 2.5: Structure of a convolutional neural network [Typ, 2020].

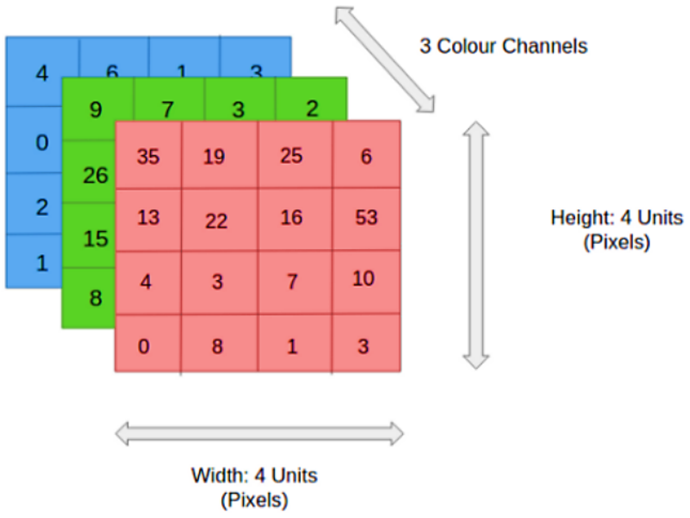


Fig 2.6: RGB image represented in matrix form [Dumoulin, 2020].

2.5.2.1 The Convolution Layer

The convolution layer is the first and one of the most fundamental layers in a convolutional neural network (CNN). A CNN may consist of one or more convolution layers, each functioning as a feature extractor by taking RGB images as input [LeCun et al., 2010].

The convolution operation involves computing the sum of element-wise multiplications between a filter (also called a convolution kernel) and small regions (or slices) of the input image matrix, as illustrated in Figure 2.9. These small regions have the same spatial dimensions as the filter. The filter slides over the input image, overlapping different regions until it covers the entire image, as shown in Figure 2.8.

Each filter has a width, height, and depth, where the depth corresponds to the number of filters applied and determines the number of output feature maps. The filter size is a user-defined parameter. The values or weights of these filters are initially set randomly and are learned and updated during the training phase [22, 23].

For example, a convolutional layer with a filter size of $3 \times 3 \times 64$ indicates that there are 64 filters, each of size 3×3, producing 64 corresponding output feature maps. The filter moves across the input image in discrete steps defined by a parameter called stride. The stride controls the step size of the filter's movement; larger stride values produce smaller output feature maps by reducing spatial dimensions [LeCun et al., 2010].

The result of the convolution operation is a set of new images called feature maps, which are spatially reduced representations of the original image emphasizing distinctive features [22, 23, 37]. As the number of convolutional layers increases, the spatial size of the feature maps tends to decrease. To counteract excessive reduction, a technique called zero padding is often applied, where additional pixels (typically zeros) are added around the input image borders to maintain the output size [Wikipedia contributors, 2020].

Early convolutional layers tend to detect simple, primary features such as edges and textures, whereas deeper layers extract more complex, abstract, or hidden features [LeCun et al., 2010].

Finally, an activation function is applied to the output of each convolutional layer to introduce non-linearity into the network. The most commonly used activation function is the Rectified Linear Unit (ReLU), which accelerates training without compromising the network's performance [4, 24].

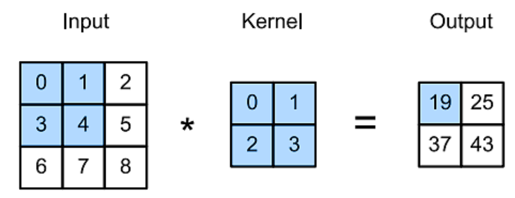


Fig 2.7: Convolution Operation [Zhang et al., 2020].

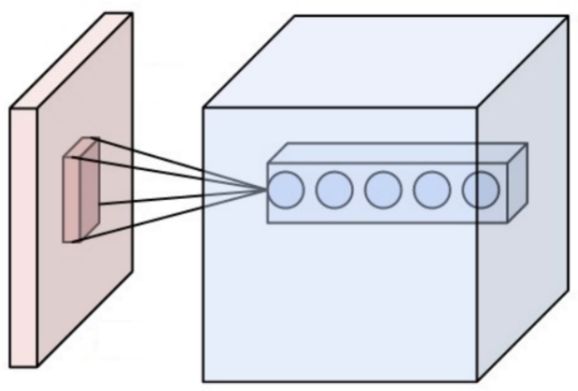


Fig 2.8: Convolution Operation [Con, 2020].

2.5.2.2 The Pooling Layer

Pooling layers typically follow convolutional layers in a CNN architecture. These layers operate on the feature maps produced by the convolutional layers by dividing each feature map into smaller regions or slices matrices of a user defined size and then traversing all such slices. The size of these slices and the step size (stride) used to move across the feature map are set by the user. Generally, small slice sizes are preferred $(2 \times 2 or 1 \times 1)$ because larger slices would excessively reduce the spatial dimensions of the feature maps, leading to a substantial loss of information.

Pooling then aggregates values within each slice by either taking the average or the maximum value. When the average is computed, this operation is called Average Pooling; when the maximum value (representing the strongest activation) is selected, it is called Max Pooling [LeCun et al., 2010]. Max pooling is the most commonly used pooling technique.

Figure 2.9 illustrates the Max Pooling operation on a 4×4 image matrix using a 2×2 slice with a stride of 2. The maximum value in the first slice (highlighted in orange) is 6. For the subsequent slices, the maximum values obtained are 8, 3, and 4, respectively. After this operation, the resulting feature map is reduced to a size of 2×2.

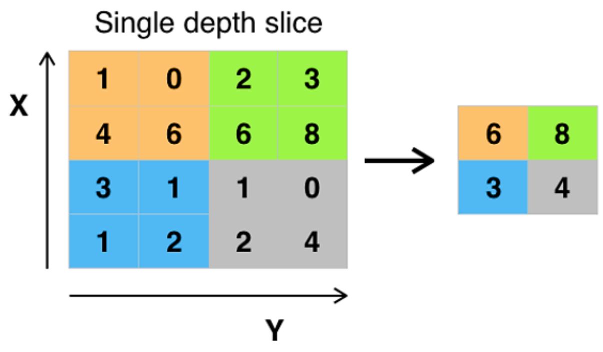


Fig 2.9: MaxPooling [Max, 2020].

The objective of the pooling operation is to reduce the spatial dimensions (width and height) of each feature map, thereby decreasing the number of parameters in the convolutional neural network and reducing its computational cost. Pooling helps eliminate redundant or noisy information from the feature maps and enhances the network's robustness to variations in the orientation and position of features within images, creating a form of translation invariance [22, 23].

2.5.2.3 The Flattening Operation and Fully Connected Layer

The outputs (matrices) obtained after a series of convolutional and pooling operations are transformed into a one-dimensional vector through a process called flattening. This operation aggregates all the extracted features from the image into a single continuous vector (see Figure 2.10). This vector is then passed to a multilayer perceptron composed of fully connected layers, which ultimately determines the class membership of the input image [22, 23].

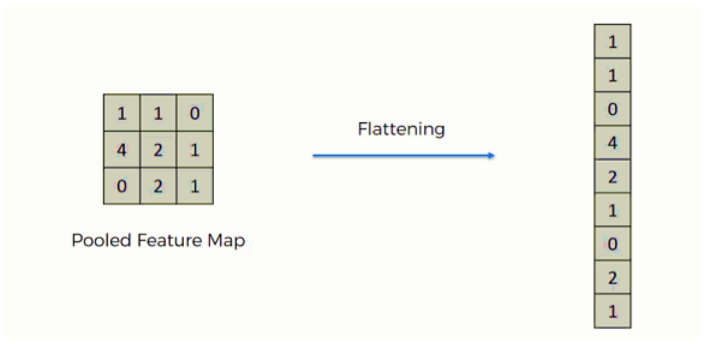


Fig 2.10: The Flattening Operation [SuperDataScience Team, 2020].

2.5.2.4 Padding

In CNNs, padding is an important parameter that adds extra pixels around the edges of the input image to help preserve its spatial dimensions. By maintaining border information, padding enables the model to better estimate the spatial size of the output feature maps. There are several padding techniques, including same padding (also known as identical padding), valid padding, and zero padding. Among these, zero padding is the most widely used due to its simplicity and computational efficiency. Zero padding involves adding zeros evenly around the border of the input matrix. This technique has been successfully employed in high-performance models such as AlexNet [Malik, 2020].

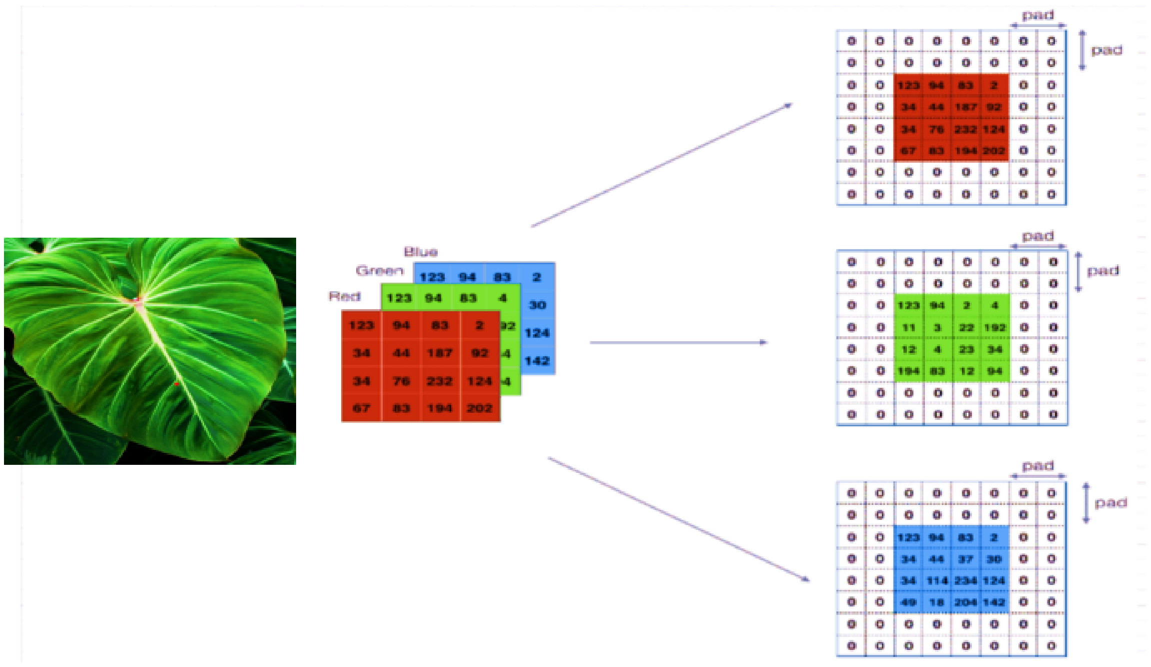


Fig 2.11: Padding example.

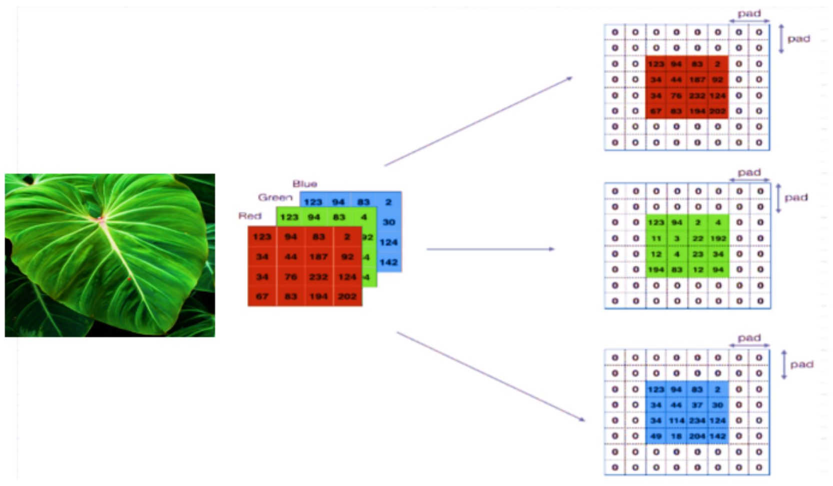


Fig 2.12: With Padding, Stride = 2.

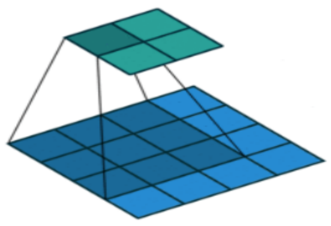


Fig 2.13: No Padding, Stride = 1.

2.5.3 The Advantage of Convolutional Neural Networks

One of the primary advantages of convolutional neural networks (CNNs) is weight sharing, meaning that each neuron (filter) within a convolutional layer uses the same set of weights. This significantly reduces the number of parameters, memory requirements, and speeds up the learning process [Wikipedia contributors, 2020]. In contrast, multilayer perceptrons assign distinct weights to every input connection, which increases the number of parameters and computational cost; as image size grows, this problem becomes more severe [Wikipedia contributors, 2020].

Another key benefit of CNNs over multilayer perceptrons is their spatial invariance. CNNs can detect features regardless of their orientation or position in the image, making them robust to spatial translations of the input [LeCun et al., 2010].

2.5.4 The Effectiveness of CNNs in Image Classification

Convolutional Neural Networks are especially well-suited for image classification tasks due to their ability to automatically and efficiently learn spatial hierarchies of features from raw input images. Unlike traditional machine learning algorithms that rely on manual feature engineering, CNNs learn low-level features such as edges and textures in early layers, and more abstract, high-level features such as shapes and objects in deeper layers [LeCun et al., 2015], [Krizhevsky et al., 2012]. The effectiveness of CNNs in image classification stems from several key characteristics:

- **Local Receptive Fields:** CNNs process small regions of the image at a time, enabling the capture of local spatial patterns essential for visual understanding [Goodfellow et al., 2016].
- **Weight Sharing:** Shared weights across spatial regions reduce the number of trainable parameters, improving computational efficiency and reducing the risk of overfitting [LeCun et al., 2015].
- **Translation Invariance:** Pooling layers and convolutional filters confer robustness to the position of features within the image [Krizhevsky et al., 2012].
- **Automatic Feature Learning:** CNNs eliminate the need for handcrafted features by learning the most discriminative features directly from raw image data [Simonyan and Zisserman, 2014].
- **Depth and Hierarchical Representation:** The layered architecture of CNNs facilitates learning complex, hierarchical features that enhance classification accuracy across diverse image datasets [LeCun et al., 2015], [He et al., 2016].

These properties make CNNs highly effective for large-scale image classification with applications in medical diagnostics, object detection, satellite imagery analysis, and more.

2.5.5 Overfitting and Underfitting in CNN Models

New plant disease detection models using image processing aim to minimize false positives and false negatives. Locally developed models have shown promising predictive performance, which can be further improved by leveraging cloud-based services and access to larger datasets, enabling easier fine-tuning. Enhanced preprocessing techniques have increased accuracy without overfitting, making these models practical beyond academic research by providing farmers with tools to improve crop yields.

However, challenges remain: issues in data acquisition or the use of overly cleaned datasets can cause premature specificity, while real-world images often vary due to different angles, distances, and environmental conditions. The performance of off-the-shelf solutions in these variable conditions remains unclear, although baseline models offer a useful starting point.

In a representative dataset, at least 20 images each of Common Rust and healthy plants were used. Noise was removed, and low-resolution images discarded. Leaf surfaces were captured with handheld cameras at angles between 15Â° and 45Â°. Common Rust images displayed characteristic round, dark red spots, while healthy leaves appeared dark green.

Three types of CNN architectures were implemented using the Keras API, with hyper-parameter searches for activation functions, optimizers, learning rates, and early stopping criteria. The impact of image format (PNG) and the use of 4-channel RGB data was minimal [Yasin and Fatima, 2023]. Incorporating dropout layers and max-pooling improved model performance; however, all models failed overfitting tests, with accuracy scores below 0.84 and similar misclassification patterns observed in confusion matrices.

2.6 State of the Art for Plant Disease Detection Using CNNs

As of 2025, the field of plant disease detection using Convolutional Neural Networks (CNNs) has made significant advancements, with research focusing on improving accuracy, computational efficiency, and real-time applicability. The following overview highlights the current state-of-the-art approaches:

Table 2.1: Performance comparison of plant disease detection models

Model	**Accuracy**	**Key Features**	**Deployment**	**Reference**
FourCropNet	Up to 99.7%	Residual blocks, attention mechanisms, multi-crop detection	Scalable across various crops	[Khandagale and Others, 2025]
14-DCNN	99.97%	14-layer CNN, data augmentation, optimization	High-accuracy applications	[Palanisamy and Kandasamy, 2022]
Lite-MDC	94.14–99.78%	Depthwise convolutions, lightweight design	Real-time, edge devices	[lit, 2024]
PlantXViT	93.55–98.33%	CNN-ViT hybrid, explainability tools	IoT, smart agriculture	[Thakur et al., 2022a]
ICVT	Up to 99.94%	Inception + Transformer architecture	Accurate, interpretable models	[icv, 2022]

These models represent the state-of-the-art in plant disease detection using CNNs, each offering unique advantages in terms of accuracy, efficiency, and applicability to various deployment scenarios.

2.7 Conclusion

This chapter provides a comprehensive overview of data collection, model deployment, and training considerations for machine learning-based plant disease detection. It discusses convolutional neural networks, unsupervised learning, and reinforcement learning techniques, alongside rigorous data collection protocols. The chapter also reviews state-of-the-art imaging modalities and effective training methods. Furthermore, it examines practical implementations of in-field disease detection using mobile and edge computing platforms, presenting a deployment roadmap that balances platform capabilities with operational constraints. Detailed discussions on the experimental pipeline, data augmentation strategies, and ensemble model architectures will be presented in the following chapter.

Chapter 3

Implementation and Experimental Analysis

3.1 Introduction

In this chapter, we present the design of our system and evaluate its performance in detecting agricultural diseases from leaf images. We utilized a dataset containing images of both healthy and diseased leaves. Deep learning models were developed to identify disease-specific patterns, enabling the classification of various types of plant illnesses. Additionally, we describe the implementation tools and the computing platform used for our experiments.

Finally, we present the results of our study, demonstrating that the trained models achieve high accuracy in plant disease detection. The performance was validated using the PlantVil-lage dataset, one of the largest and most widely used datasets for plant disease classification.

3.2 Development Environment

This section provides a brief overview of the software and hardware specifications used in this study. Python was chosen for its versatility and comprehensive machine learning librar-ies, while PyTorch was selected for its dynamic computational graph capabilities and user-friendly tensor operations, facilitating efficient neural network implementation. The hard-ware setup includes high-performance GPUs to accelerate both model training and inference processes.

Materiels / Software	Parameters
Processor	i9 Intel 13^{e} generation-13900KF 3.00 GHz
Memory capacity	128 Go
Graphic processing unit (GPU)	GeForce RTX 4080
Windows	10 Professionnel
Cuda	12.1
Python	3.11
PyTorch	2.1

Table 3.1: System Specifications

3.2.1 PyTorch

PyTorch is an open-source deep learning framework developed by Meta (formerly Facebook). It is widely used in both research and industry to build and train neural networks for various AI applications like image classification, natural language processing, and more.

3.2.1.1 Key Components of PyTorch

PyTorch provides a rich set of components designed to facilitate the development and training of deep learning models efficiently. The following are some of the essential components commonly used in PyTorch workflows:

Component	Description
`torch.Tensor:`	A multi-dimensional array like NumPy, but with GPU support
`torch.nn:`	Module for building neural networks with layers like `Linear`, `Conv2d`, etc.
`torch.optim:`	Optimization algorithms (e.g., SGD, Adam) for updating model weights
`Autograd:`	Automatic differentiation engine (used in backpropagation)
`DataLoader:`	Helps load and batch data efficiently for training

Table 3.2: Key PyTorch Components and Their Descriptions

3.2.1.2 Utility of PyTorch

PyTorch is a Python-based framework that facilitates deep learning development through its clear syntax, simplified API, and efficient debugging capabilities. Its core data structure, the tensor which is similar to NumPy arrays offers enhanced functionality such as GPU ac-celeration, distributed computing across multiple devices, and dynamic computation graph tracking. These features are essential for modern deep learning applications. PyTorch has proven effective in professional environments, enabling large-scale, real-world tasks [PyTorch Team, 2024].

Two key features make PyTorch especially relevant for deep learning:

1. **Accelerated computation using GPUs**, which can achieve speedups up to 50

times faster compared to equivalent computations on a Central Processing Unit (CPU).

2. **Support for numerical optimization over general mathematical expressions**, which is critical for training deep learning models.

It is important to note that these features are valuable not only for deep learning but also for scientific computing in general. Indeed, PyTorch can be considered a high-performance li-brary with built-in optimization capabilities for scientific computation in Python [Paszke, 2019].

3.2.1.3 Applications

PyTorch is widely used in various domains, including but not limited to:

- Computer Vision: Image classification, object detection, and segmentation
- Natural Language Processing: Machine translation, sentiment analysis, and text generation
- Reinforcement Learning: Training agents for decision-making tasks
- Scientific Computing and Research: High-performance numerical simulations and modeling

3.3 System Architecture

The diagram illustrates a comprehensive CNN-based workflow for plant leaf disease classification. The top section represents the training process, where plant leaf images exhibiting various diseases undergo preprocessing. This preprocessing likely includes resizing images to 224×224 pixels, normalizing pixel values to a 0-1 scale, and applying data augmentation techniques such as rotations and flips. The preprocessed images are then fed into the CNN architecture either SimpleCNN or DeepCNN which processes them through multiple convolutional and pooling layers. The model trains over N epochs, with each epoch representing a complete pass through the training dataset. During training, the model continuously computes loss values and updates its weights via backpropagation to improve prediction accuracy.

The middle section depicts the validation workflow, where a separate set of leaf disease images undergoes identical preprocessing and is evaluated by the same CNN model. This validation phase calculates loss and accuracy metrics without updating model weights, providing an unbiased assessment of the model's generalization capability to unseen data. To mitigate overfitting, the system employs early stopping by saving the model version that achieves the best validation accuracy.

The bottom section corresponds to the final testing phase, where the saved best performing model is evaluated on an entirely separate test dataset. This evaluation encompasses multiple performance metrics beyond accuracy, including loss, precision, recall, and F1 score, offering a comprehensive understanding of the model's effectiveness in real-world scenarios. The blue arrows connecting validation results to model saving and subsequently to testing illustrate the critical feedback loop that ensures only the optimal model advances to final evaluation.

This well-structured pipeline-with clear separation between training, validation, and testing phases-follows standard best practices in deep learning for image classification tasks and is specifically optimized for plant disease diagnosis, as depicted in Figure 3.1.

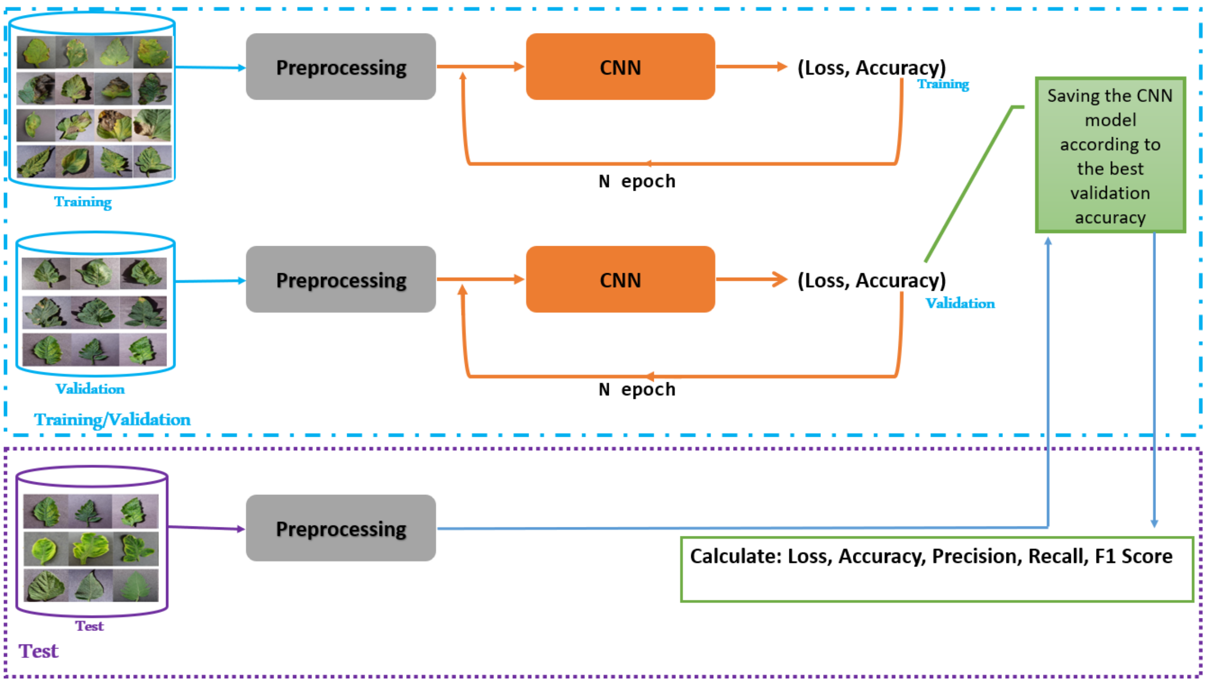


Fig 3.1: Our system of plant leaf diseases using convolutional neural network.

3.4 Dataset Selection and Preprocessing

Effective development of machine learning models for plant disease detection heavily relies on the availability of high-quality, annotated datasets and appropriate preprocessing tech-niques. This section outlines the rationale behind the choice of datasets, with a focus on the widely-used PlantVillage dataset, and details the preprocessing steps necessary to prepare the data for training convolutional neural networks.

Selecting an appropriate dataset is crucial to ensure model robustness, generalizability, and real-world applicability. The PlantVillage dataset is a comprehensive collection of annotated images encompassing a diverse range of plant species and disease classes. It provides stand-ardized RGB images, facilitating consistent input dimensions and enabling efficient training.

Preprocessing transforms raw images into a format suitable for model ingestion, including resizing, normalization, and data augmentation techniques such as rotations, flips, and color jittering. These steps not only standardize the input data but also enhance model perfor-mance by increasing dataset variability and reducing overfitting.

Together, dataset selection and preprocessing form the foundational steps that directly im-pact the accuracy and reliability of the plant disease detection models developed in this study.

3.4.1 Description of Selected Dataset (PlantVillage)

The PlantVillage database is a pivotal resource for research and the development of diagnos-tic tools for plant diseases, combining artificial intelligence techniques with expertly anno-tated images to improve the management of agricultural crops globally. This dataset supports the detection and control of a broad spectrum of biotic plant diseases, including bacterial, fungal, viral, and nematode infections.

The database contains over 54,323 annotated images across 14 plant species, categorized into 38 classes representing various diseases and healthy conditions. Additionally, there is a separate class for background images without foliage. The images are RGB and standardized to a resolution of 224 × 224 pixels, making the dataset an essential foundation for developing and validating machine learning models for automated plant disease diagnosis [30].

Dealing with an unbalanced dataset where certain classes have significantly more instances than others poses a major challenge. To prevent model bias, it is crucial to accurately iden-tify minority classes and assign equal importance to all classes during training. Balancing the training set often requires data augmentation techniques, which involve generating synthetic data or variations of existing samples.

In this study, the images in the dataset were divided into three subsets for training the CNNs model:

- **Training set (80%)**: Used to enable the model to learn underlying patterns and fea-ture representations within the data.

Plant	Disease	Train	Validation	Test
Apple	Apple scab	630	621	629
	Black rot	620	615	621
	Cedar apple rust	275	258	268
	Healthy	1645	1603	1642
Blueberry	Healthy	1201	1487	1501
Cherry	Healthy	683	116	846
	Powdery mildew	1052	1048	1051
	Cercospora leaf spot Gray leaf spot	513	506	512
Corn	Common rust	1192	1188	1191
	Healthy	1162	1142	1156
	Northern Leaf Blight	985	974	978
Grape	Black rot	1180	1142	1177
	Esca (Black Measles)	1383	1359	1382
	Healthy	423	417	420
	Leaf blight	1076	1061	1070
Orange	Haunglongbing (Citrus greening)	5507	4238	5506
Peach	Bacterial spot	2296	2286	2297
	Healthy	360	345	355
Pepper	Bacterial spot	997	985	989
	Healthy	1477	1448	1471
Potato	Early blight	1000	987	990
	Healthy	152	135	146
	Late blight	1000	987	990
Raspberry	Healthy	371	365	369
Soybean	Healthy	5090	5087	5088
Squash	Powdery mildew	1835	1817	1832
Strawberry	Healthy	456	446	455
	Leaf scorch	1109	1102	1106
	Bacterial spot	2127	2105	2109
	Early blight	1000	987	990
	Healthy	1590	1581	1591
	Late blight	1908	1897	1904
Tomato	Leaf Mold	952	948	951
	Septoria leaf spot	1769	1742	1771
	Spider mites (Two-spotted spider mite)	1676	1664	1673
	Target Spot	1404	1390	1392
	Tomato mosaic virus	373	365	372
	Tomato Yellow Leaf Curl virus	5357	5355	5356

- **Validation set (10%)**: Employed during training to monitor model performance, tune hyperparameters, and mitigate overfitting by assessing the model's generalization ability on unseen data.
- **Test set (10%)**: Reserved for evaluation after training and validation are complete, providing an independent and unbiased measure of the model's predictive performance on entirely new inputs, thus indicating its generalization capacity.

3.4.2 Image Preprocessing Techniques

Image preprocessing refers to a series of systematic operations applied to raw images before feeding them into machine learning or computer vision models. The main goals are to en-hance image quality, normalize data, and make the images consistent for improved model performance and accuracy. The common preprocessing techniques used in this work include:

3.4.2.1 Data Augmentation

Data augmentation enriches the training dataset by artificially generating new samples, increasing the diversity of features the model can learn. The applied techniques fall into two categories:

- **Photometric modifications**: Adjustments in hue, saturation, contrast, and luminance.
- **Geometric transformations**: Operations such as rotation, translation, resizing, shear-ing, zooming, and flipping.

In this study, three augmentation techniques-horizontal flipping, vertical flipping, and rotation were applied to enhance data diversity and improve model robustness against spatial variations.

3.4.2.2 Resizing

Resizing ensures all input images have uniform dimensions before being fed into the neural network. Most convolutional neural networks (CNNs), including AlexNet and ResNet, require fixed input sizes such as 224×224 pixels.

3.4.2.3 Normalization

Normalization standardizes pixel values, typically scaling them to a fixed range such as [0, 1] or [-1, 1]. This process, often implemented as zero-mean and unitvariance scaling, accelerates convergence, prevents numerical instability, and stabilizes training.

3.4.3 Implementation of Data Pipelines Using PyTorch

The following describes the setup of training, validation, and testing data pipelines using TensorFlow's Keras ImageDataGenerator, a utility that loads images from directories and applies real-time data augmentation.

- **Batch size**: Set to 16, balancing memory efficiency and training speed. Smaller batches can improve generalization but may slow training, while larger batches may converge faster but risk poorer generalization.
- **Image Data Generator**: Provides real-time data augmentation, which enhances model robustness and reduces overfitting by creating modified versions of training images dynamically.

For the training data generator (train_datagen), the following parameters were applied:

- rescale=1./255: Normalizes pixel values from [0, 255] to [0, 1].
- rotation_range=10: Randomly rotates images up to Â ± 10 degrees, improving orientation invariance.
- shear_range=0.2: Applies random shearing transformations simulating different viewing angles.
- zoom_range=0.2: Randomly zooms into images, aiding scale invariance.
- horizontal_flip=True: Randomly flips images horizontally, appropriate here since leaf orientation typically does not alter disease characteristics.

The training data generator loads images from the directory ./output/train, resizing them to 224×224 pixels, using the batch size of 16, and producing one-hot encoded labels for multi-class classification.

For **validation data**, a similar generator (val_datagen) was configured, although typically data augmentation is not applied to validation sets to ensure consistent evaluation conditions. Here, identical augmentation parameters were used, which may affect the accuracy of validation performance estimates.

For **test data**, the generator uses only rescaling (normalization) without augmentation, ensur-ing evaluation on unmodified images for an accurate assessment of real world performance.

```
        batch_size=16
        from tensorflow.keras.preprocessing.image import
           ImageDataGenerator
        train_datagen=ImageDataGenerator(rescale=1./255,
           rotation_range=10, shear_range=0.2, zoom_range
           =0.2, horizontal_flip=True)
        train_generator=train_datagen.flow_from_directory(
        './output/train',
        #R'C:\Malik\LI3C\Projet de recherche\PNR 2022\
           databases\taiwan\taiwan\data augmentation\Train',
        target_size=(224,224 ),
        batch_size=batch_size,
        classes = classes_train,
        class_mode='categorical'
        )
        val_datagen=ImageDataGenerator(rescale=1./255,
           rotation_range=10, shear_range=0.2, zoom_range
           =0.2, horizontal_flip=True)
        val_generator=val_datagen.flow_from_directory(
        './output/val',
        target_size=(224, 224),
        batch_size=batch_size,
        classes = classes_train,
        class_mode='categorical'
        )
        val_datagen=ImageDataGenerator(rescale=1/255)
        test_generator=val_datagen.flow_from_directory(
        './output/test',
        target_size=(224, 224),
        batch_size=batch_size,
        classes = classes_train,
        class_mode='categorical'
        )
```

3.5 CNN Model Development

This section presents the development of convolutional neural network (CNN) models tailored for plant disease detection. CNNs are well-suited for image classification tasks due to their ability to automatically extract hierarchical spatial features from raw image data. We explore two architectures: a custom Deep CNN and a simpler CNN model, both implement-ed using PyTorch.

The following subsections detail the architectures, layer configurations, and training methodologies employed in these CNN models.

3.5.1 Deep CNN

The following code defines a custom deep Convolutional Neural Network (CNN) architecture implemented in PyTorch. Below is a detailed analysis of its structure and functionality:

The DeepCNN class inherits from nn. Module, which is the base class for all neural network modules in PyTorch. This inheritance provides essential functionality such as parameter tracking and utilities to move the model between CPU and GPU.

Within the _init_method, the model is initialized with num_classes as a parameter, which determines the number of output nodes in the final classification layer. (Note: This parameter is not explicitly used in the visible portion of the code.)

The network is organized into two main components: **features** and **classifier**. This design pattern is common in CNN architectures where the **features** section is responsible for extracting hierarchical representations from images, and the **classifier** interprets these features to produce class predictions.

The **features** section is implemented as an nn.Sequential container, facilitating a clean and modular definition of the network layers. It consists of four convolutional blocks, each following a similar structure:

1. **First block:**
   - nn.Conv2d(3, 64, 3, padding=1): A 2D convolutional layer accepting 3 input channels (RGB image), producing 64 output feature maps using a 3×3 kernel with padding of 1 pixel on all sides to maintain spatial dimensions.

- nn.ReLU(): Applies the Rectified Linear Unit activation, introducing non-linearity.
- Another convolutional layer with 64 input and 64 output channels.
- Another nn.ReLU() activation.
- nn.MaxPool2d(2): Max pooling with a 2×2 window, reducing spatial dimensions by half to 112×112 (assuming input images of size 224×224).

2. **Second block:** Similar to the first block but increases the number of feature channels from 64 to 128, with spatial dimensions further reduced to 56×56.
3. **Third block:** Increases feature channels from 128 to 256, with spatial dimensions reduced to 28v28.
4. **Fourth block:** Increases feature channels from 256 to 512, producing feature maps of size 14×14.

The classifier section begins with:

- nn.Dropout(0.5): Randomly zeroes 50% of the inputs during training, serving as a regularization technique to reduce overfitting.
- nn.Linear(512 * 14 * 14, 512): A fully connected layer that flattens the feature maps (512 channels × 14 × 14 spatial size) and reduces them to a 512-dimensional feature vector.
- nn.ReLU(): Applies another non-linear activation.

```
    def __init__(self, num_classes):
    super(DeepCNN, self).__init__()
    self.features = nn.Sequential(
    nn.Conv2d(3, 64, 3, padding=1), nn.ReLU(),
    nn.Conv2d(64, 64, 3, padding=1), nn.ReLU(),
    nn.MaxPool2d(2),  # (64, 112, 112)

    nn.Conv2d(64, 128, 3, padding=1), nn.ReLU(),
    nn.Conv2d(128, 128, 3, padding=1), nn.ReLU(),
    nn.MaxPool2d(2),  # (128, 56, 56)

    nn.Conv2d(128, 256, 3, padding=1), nn.ReLU(),
    nn.Conv2d(256, 256, 3, padding=1), nn.ReLU(),
    nn.MaxPool2d(2),  # (256, 28, 28)

    nn.Conv2d(256, 512, 3, padding=1), nn.ReLU(),
    nn.Conv2d(512, 512, 3, padding=1), nn.ReLU(),
    nn.MaxPool2d(2),  # (512, 14, 14)
    )
    self.classifier = nn.Sequential(
    nn.Dropout(0.5),
    nn.Linear(512 * 14 * 14, 512),
    nn.ReLU(),
    nn.Linear(512, num_classes)
    )

    def forward(self, x):
    x = self.features(x)
    x = x.view(x.size(0), -1)
    return self.classifier(x)
```

3.5.2 Simple CNN

The SimpleCNN class defines a straightforward convolutional neural network architecture implemented using PyTorch. It is designed for image classification tasks and accepts an input of RGB images, producing predictions over a specified number of output classes (num_classes).

Architecture Overview

- **Inheritance:**

  The class inherits from nn.Module, the base class for all neural network models in PyTorch. This provides the infrastructure needed for parameter management and forward propagation.

- **Features Extraction (self.features):**

  This sequential container consists of three convolutional blocks. Each block includes:

  – A **2D convolutional layer (nn.Conv2d)** that applies learnable filters to extract spatial features from input images.

- A **ReLU activation (nn.ReLU)** introducing non-linearity, allowing the model to learn complex patterns.
- A **max pooling layer (nn.MaxPool2d)** that downsamples the spatial dimensions by a factor of 2, reducing computational load and providing translation invariance.

Specifically, the blocks are:

- **First block:** Takes the 3-channel RGB input and outputs 32 feature maps of size 224×224 (same padding). After max pooling, the spatial size is reduced to 112×112.
- **Second block:** Converts 32 feature maps to 64, maintaining spatial size 112×112 before max pooling, then reducing to 56×56.
- **Third block:** Expands 64 feature maps to 128, with spatial dimensions reduced from 56×56 to 28×28 after pooling

• **Classification Head (self.classifier):**

This section transforms the extracted features into final class predictions:

- nn.Dropout(0.5): Randomly zeroes 50% of inputs during training to prevent overfitting.
- nn.Linear(128 * 28 * 28, 256): Fully connected layer that flattens the feature maps (128 channels × 28 height × 28 width = 100,352 features) into a vector of size 256.
- nn.ReLU(): Activation function adding non-linearity.
- nn.Linear(256, num_classes): Final fully connected layer mapping to the output classes.

Forward Pass (forward method)

- The input tensor × (a batch of images) is passed through the feature extraction layers. × The resulting tensor is flattened into a 2D shape (batch_size, features) using x.view(x.size(0), -1) to prepare for the fully connected layers. × The flattened features are passed through the classifier to produce the final output logits corresponding to each class.

```
import torch
import torch.nn as nn

class SimpleCNN(nn.Module):
def __init__(self, num_classes):
super(SimpleCNN, self).__init__()
self.features = nn.Sequential(
nn.Conv2d(3, 32, 3, padding=1),  # (32, 224, 224)
nn.ReLU(),
nn.MaxPool2d(2),                # (32, 112, 112)

nn.Conv2d(32, 64, 3, padding=1),# (64, 112, 112)
nn.ReLU(),
nn.MaxPool2d(2),                # (64, 56, 56)

nn.Conv2d(64, 128, 3, padding=1),# (128, 56, 56)
nn.ReLU(),
nn.MaxPool2d(2),                # (128, 28, 28)
)
self.classifier = nn.Sequential(
nn.Dropout(0.5),
nn.Linear(128 * 28 * 28, 256),
nn.ReLU(),
nn.Linear(256, num_classes)
)

def forward(self, x):
x = self.features(x)
x = x.view(x.size(0), -1)
return self.classifier(x)
```

3.6 Using Pre-trained Models (DenseNet121, AlexNet, and VGG19) for Transfer Learning

3.6.1 DenseNet121

DenseNet-121 is a deep convolutional neural network consisting of 121 layers characterized by dense connectivity, where each layer receives inputs from all preceding layers. This de-sign improves feature propagation and alleviates the vanishing gradient problem. DenseNet-121 employs 1×1 and 3×3 convolutions combined with ReLU activation and transition layers that reduce feature map dimensions, resulting in high performance and computational effi-ciency [GeeksforGeeks, 2025].

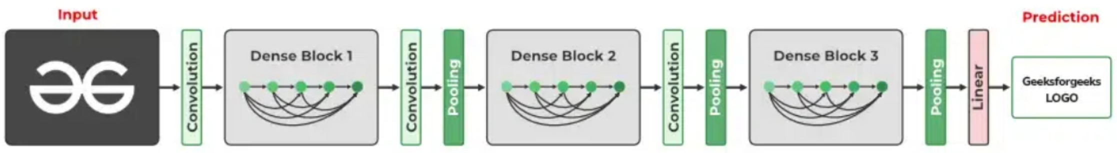


Fig 3.2: DenseNet121 architecture

3.6.2 AlexNet

Proposed by Krizhevsky et al. in 2012 [PyTorch Team, 2024], AlexNet is a pioneering convolutional neural net-work model that popularized deep learning for image recognition and transfer learning. It comprises five convolutional layers followed by three fully connected layers, utilizing ReLU activations for non-linearity and max-pooling layers to reduce spatial dimensions and pre-vent overfitting. AlexNet laid the foundation for subsequent advanced architectures such as VGG, GoogLeNet, and ResNet.

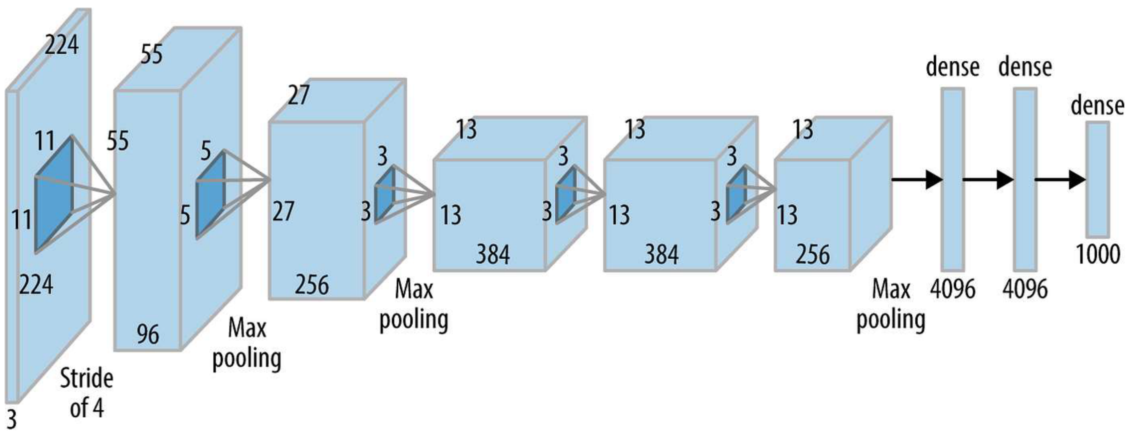


Fig 3.3: Architecture AlexNet [Krizhevsky et al., 2012].

3.6.3 VGG19

VGG-19 is a deep convolutional neural network composed of 19 layers, including 16 convo-lutional layers that consistently use 3×3 convolutional filters, ReLU activation functions, and max-pooling operations to extract and compress image features. The network ends with fully connected layers and a Softmax layer for classification. Its simple and uniform architecture has made it a benchmark model, valued for ease of understanding and implementation [GeeksforGeeks, 2025].

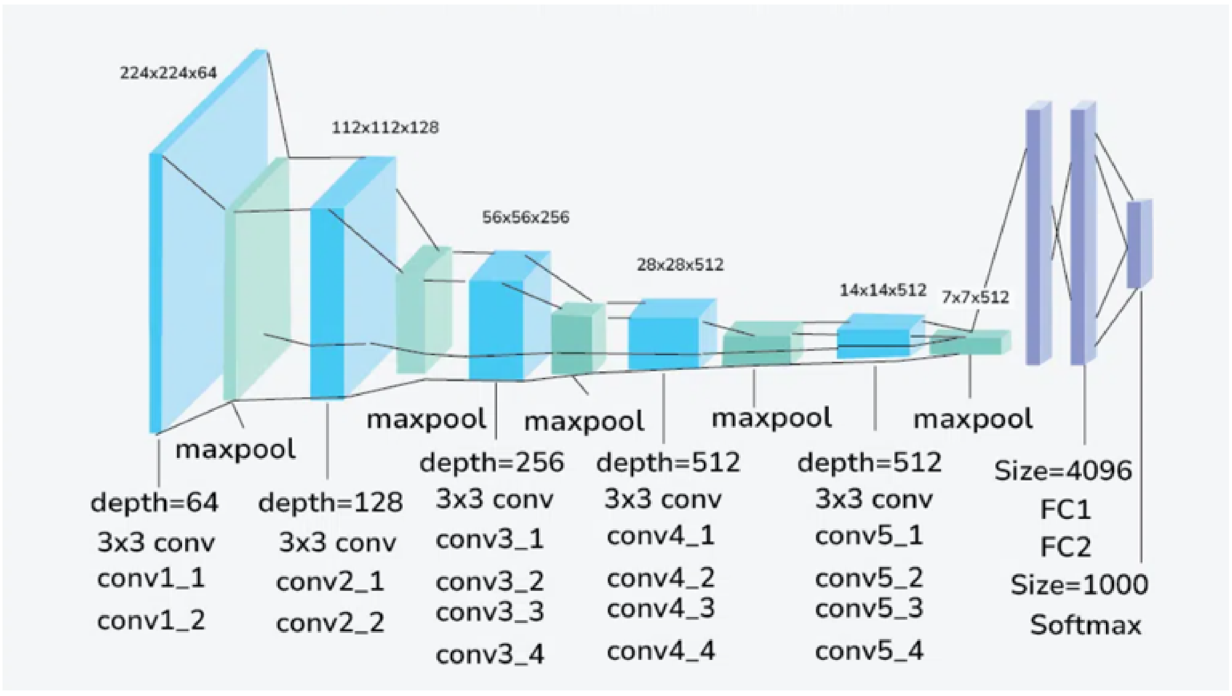


Fig 3.4: VGG19 architecture [GeeksforGeeks, 2025].

3.7 Choice of Hyperparameters

The configuration of the model depends on carefully selected hyperparameters that significantly influence the training process, model performance, and generalization capability. Table 3.3 summarizes the key hyperparameters used in this work, along with their descriptions and specific roles.

Parameter	**Value**	**Description**	**Usage**
IMG_SIZE	(224, 224)	Input image size, resized before being fed into the model.	Ensures uniform input dimensions, balancing computational cost and visual detail.
BATCH_SIZE	64	Number of training samples processed in each batch.	Affects memory usage and training stability; larger size may improve convergence speed.
epochs	1000	Number of complete passes through the training dataset.	Defines training duration; a high value may improve learning but risks overfitting.
loss	categorical_crossentropy	Measures the error between predicted and actual labels.	Appropriate for multi-class classification problems.
optimizer	Adam (lr = 0.0003, weight_decay = 0.01)	Adaptive optimization algorithm with regularization.	Dynamically adjusts learning rates; L2 regularization reduces overfitting.

Table 3.3: Hyperparameters used in this work.

These hyperparameters were selected based on prior research and empirical testing to balance training efficiency, accuracy, and generalization. For instance, the use of a high epoch count is controlled by applying L2 regularization via the optimizer's weight decay, reducing the risk of overfitting. The batch size of 64 was found to be optimal for the available hardware, providing a good trade-off between speed and memory usage.

3.8 Evaluation indices

To comprehensively assess the performance of the plant disease classification models, several evaluation metrics are employed. These metrics provide insights into different aspects of model accuracy, reliability, and robustness, especially in multi-class classification contexts [Gupta et al., 2012].

3.8.1 Accuracy

Accuracy is the most intuitive performance measure, defined as the ratio of correctly predicted instances to the total number of instances evaluated. It reflects the overall effectiveness of the model in classifying samples correctly.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

where TP is true positives, TN is true negatives, FP is false positives, and FN is false negatives.

3.8.2 Precision

Precision measures the proportion of correctly predicted positive observations among all predicted positives. It indicates how many of the positive predictions were actually correct, reflecting the model's ability to avoid false positives.

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

3.8.3 Recall (Sensitivity)

Recall, or sensitivity, quantifies the proportion of actual positives that were correctly identified by the model. It captures the model's ability to detect all relevant cases and is particularly important when missing positive instances is costly.

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

3.8.4 Score

The F1 Score is the harmonic mean of precision and recall, providing a balanced measure that accounts for both false positives and false negatives. It is especially useful when the class distribution is imbalanced.

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.4}$$

These evaluation metrics collectively offer a robust framework to evaluate classification models, ensuring a nuanced understanding of their predictive performance beyond simple accuracy.

3.9 Performance Evaluation and Results

This section presents a detailed analysis of the performance of the developed convolutional neural network models, as well as a comparison with existing state-of-the-art architec-tures and their transfer learning variants. The evaluation encompasses multiple datasets (training, validation, and test sets) and employs several key performance metrics, including loss, accuracy, precision, recall, F1 score, processing time, number of trainable parameters, and storage requirements.

The analysis begins with a comprehensive comparison between the proposed Deep CNN and Simple CNN models, highlighting their respective strengths in terms of predictive accuracy and computational efficiency. Subsequently, the performance of these models is bench-marked against prominent pre-trained networks such as DenseNet121, AlexNet, and VGG19, along with their transfer learning adaptations.

This evaluation framework provides critical insights into the trade-offs between model complexity, resource consumption, and classification performance, thereby guiding the selection of an optimal model tailored to specific application requirements and deployment environments.

3.9.1 Performance of Deep CNN and Simple CNN

Table 3.4 compares the performance of two convolutional neural network models, **Deep CNN (CNN1)** and **Simple CNN (CNN2)**, across the training, validation, and test datasets. Multiple evaluation metrics are used, including loss, accuracy, precision, recall, F1 score, processing time, number of trainable parameters, and storage space requirements. This com-parison aims to identify which model is better suited for deployment depending on whether performance or efficiency is prioritized.

Model	**Performance Indicators**	**Training**	**Validation**	**Test**	**Trainable Parameters (millions)**	**Storage (MB)**
Deep CNN	Loss	0.0561	0.0498	0.0177	56086119	219.09
	Accuracy	98.23	99.51	99.53		
	Precision	97.83	99.55	99.37		
	Recall	97.66	99.48	99.30		
	F1 Score	97.73	99.51	99.33		
	Time	39.68 s	10.09 s	10.08 s		
Simple CNN	Loss	0.0564	0.0440	0.0274	25793639	100.76
	Accuracy	98.14	99.33	99.26		
	Precision	97.74	99.18	99.22		
	Recall	97.60	99.04	98.94		
	F1 Score	97.66	99.10	99.07		
	Time	9.66 h	6.5 s	6.5 s		

Table 3.4: Comparison of model performance, size, and training time

In terms of performance, **Deep CNN** consistently outperforms Simple CNN across all phases. During the test phase, Deep CNN achieves an accuracy of **99.53%**, slightly higher than Simple CNN's **99.26%**. Its F1 score **(99.33%)** is also superior, indicating a better balance between precision and recall. Moreover, Deep CNN exhibits a lower test loss (0.0177 vs. 0.0274), suggesting more confident and accurate predictions. These results position Deep CNN as the more reliable model when maximum predictive performance is required.

However, Simple CNN demonstrates significant advantages in terms of efficiency. It contains only **25.79 million** trainable parameters compared to Deep CNN's **56.08 million**, making it substantially lighter. The storage space required for Simple CNN is less than half that of Deep CNN (100.76 MB vs. 219.09 MB), which is beneficial for deployment in environments with limited memory or storage, such as mobile or edge devices. Additionally, Simple CNN's inference time is shorter (6.5 seconds vs. 10.08 seconds), making it more suitable for real-time applications. Although Simple CNN's training time is considerably longer, this process is typically executed once and is less critical in many deployment scenarios.

Overall, **Deep CNN** is ideal when accuracy and performance are the highest priorities, espe-cially in critical applications like medical diagnostics or security systems. Conversely, **Simple CNN** is preferable for practical deployment where computational resources are constrained and efficiency outweighs the marginal gains in accuracy. The choice between these models should be informed by the specific requirements of the application, weighing factors such as performance, model size, and inference speed.

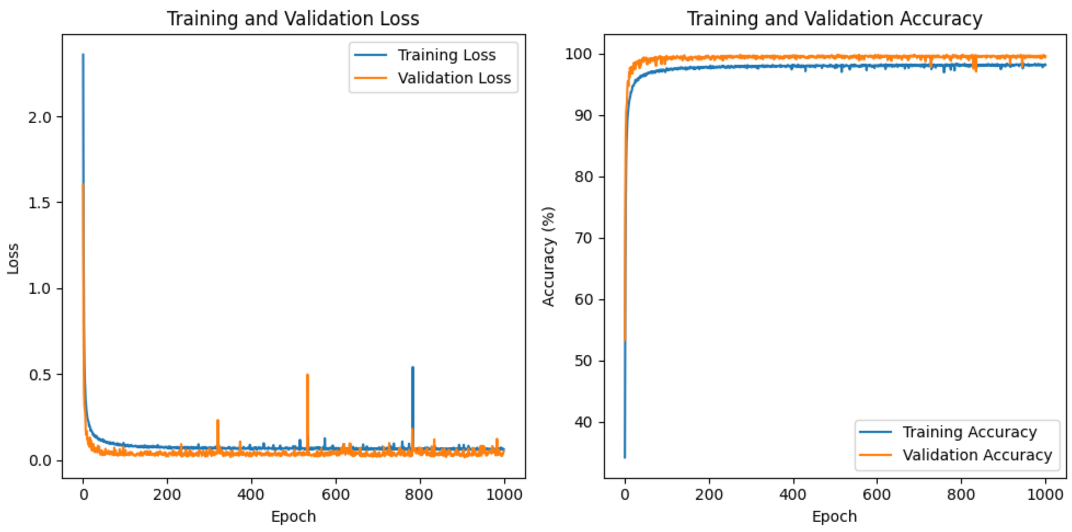


Fig 3.5: Loss and Accuracy Evolution of the Deep CNN Model.

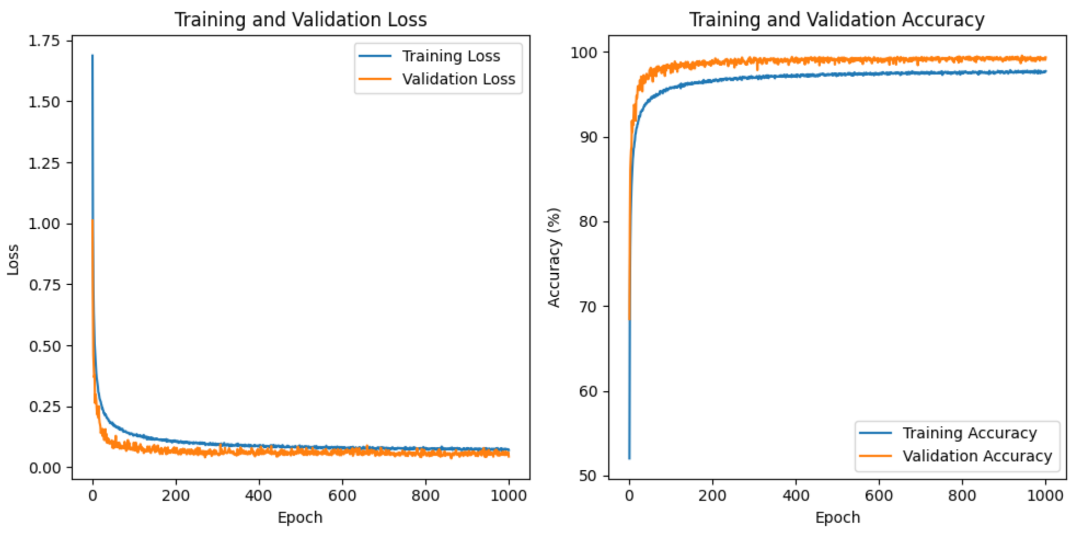


Fig 3.6: Loss and Accuracy Evolution of the SimpleCNN Model.

3.9.2 Comparison with existing models (DenseNet121, AlexNet and VGG19)

Table 3.5 presents a comprehensive comparison of multiple neural network architectures including DenseNet121, AlexNet, and VGG19 along with their transfer learning (TL) variants, as well as the previously developed Deep CNN (CNN1) and Simple CNN (CNN2) models. The comparison evaluates their performance across training, validation, and test datasets using metrics such as loss, accuracy, precision, recall, F1 score, processing time, number of trainable parameters, and storage space requirements.

Table 3.5: Comparison of CNN Architectures and Transfer Learning Variants

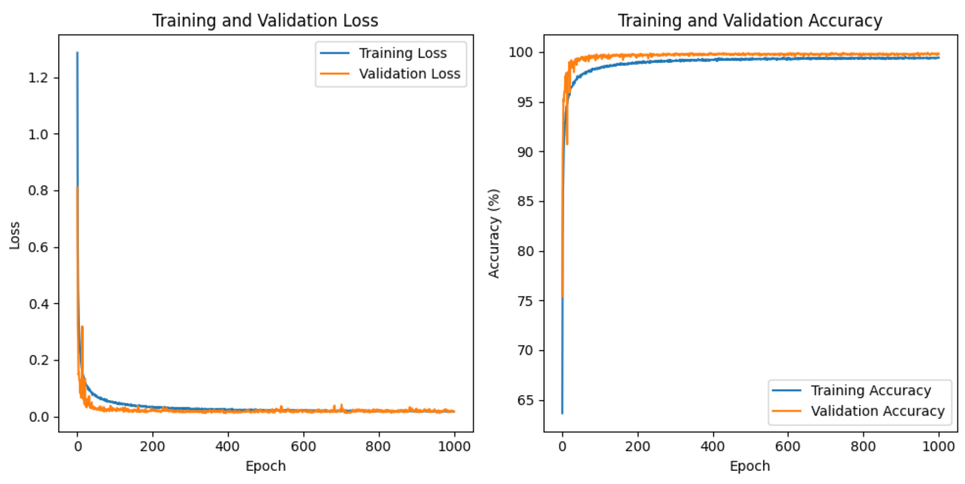
Model	**Metric**	**Train**	**Valid.**	**Test**	**Params (M)**	**Storage (MB)**
DenseNet121	Loss	0.0149	0.0149	0.0073	6.99	27.91
	Accuracy	99.51	99.51	99.86		
	Precision	99.41	99.41	99.86		
	Recall	99.44	99.44	99.82		
	F1 Score	99.43	99.43	99.84		
	Time	33.91 h	10.01 s	10.03 s		
TL DenseNet121	Loss	0.1270	0.0490	0.0462	0.54	29.88
	Accuracy	95.91	98.39	98.55		
	Precision	95.22	98.08	98.27		
	Recall	94.52	97.73	98.14		
	F1 Score	94.83	97.85	98.18		
	Time	13.68 h	9.2 s	9.4 s		
AlexNet	Loss	0.1919	0.0856	0.1214	61.10	238.68
	Accuracy	94.85	97.87	97.61		
	Precision	3.67	3.81	3.78		
	Recall	3.65	3.80	3.79		
	F1 Score	3.65	3.80	3.78		
	Time	7.28 h	11.4 s	11.4 s		
TL AlexNet	Loss	0.2505	0.1988	0.0970	4.74	28.16
	Accuracy	92.73	97.20	96.97		
	Precision	92.18	96.11	96.28		
	Recall	89.91	96.60	96.06		
	F1 Score	90.94	96.26	96.07		
	Time	7.23 h	6.5 s	6.5 s		
VGG19	Loss	0.1744	0.0579	0.0515	139.73	545.83
	Accuracy	95.94	98.37	98.49		
	Precision	95.18	98.09	98.10		
	Recall	94.56	97.93	98.05		
	F1 Score	94.82	97.99	98.05		
	Time	61.2 h	13.6 s	13.9 s		
TL VGG19	Loss	2.2685	2.0369	3.1090	104.88	488.04
	Accuracy	49.21	57.37	19.34		
	Precision	39.74	48.32	2.39		
	Recall	31.90	38.18	5.23		
	F1 Score	31.62	37.75	2.55		
	Time	49.4 h	15.1 s	15.2 s		
Deep CNN	Loss	0.0561	0.0498	0.0177	56.09	219.09
	Accuracy	98.23	99.51	99.53		
	Precision	97.83	99.55	99.37		
	Recall	97.66	99.48	99.30		
	F1 Score	97.73	99.51	99.33		
	Time	39.68 s	10.09 s	10.08 s		
Simple CNN	Loss	0.0564	0.0440	0.0274	25.79	100.76
	Accuracy	98.14	99.33	99.26		
	Precision	97.74	99.18	99.22		
	Recall	97.60	99.04	98.94		
	F1 Score	97.66	99.10	99.07		
	Time	9.66 h	6.5 s	6.5 s		

The results reveal several insights:

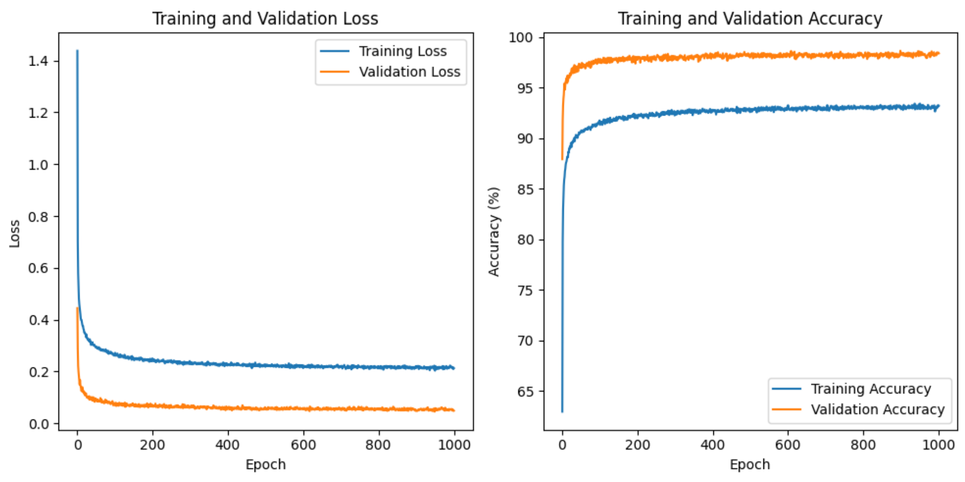
- DenseNet121 is the top-performing architecture, achieving approximately 99% across all major metrics (accuracy, precision, recall, and F1 score) during training, validation, and testing.

- Its transfer learning variant (TL DenseNet121) offers a substantial reduction in train-able parameters from nearly 7 million to approximately 544,000 while maintaining high test accuracy (98.5%). This efficiency makes it attractive for resource-constrained deployments.
- AlexNet and its transfer learning variant demonstrate moderate performance with ac curacies between 92% and 98%, though the original AlexNet shows unusually low precision and recall values, possibly due to a data labeling or evaluation issue.
- VGG19 exhibits more variable results, with high training and validation accuracy but suffers from extremely poor performance in its transfer learning form (TL VGG19), which achieves only about 19% accuracy on the test set.
- Storage requirements vary widely, with VGG19 demanding the most space (545.83 MB) and TL DenseNet121 requiring the least (29.88 MB).
- The previously developed Deep CNN and Simple CNN models perform competitively, with Deep CNN nearing DenseNet121's performance, while Simple CNN offers a good trade off between accuracy and efficiency.

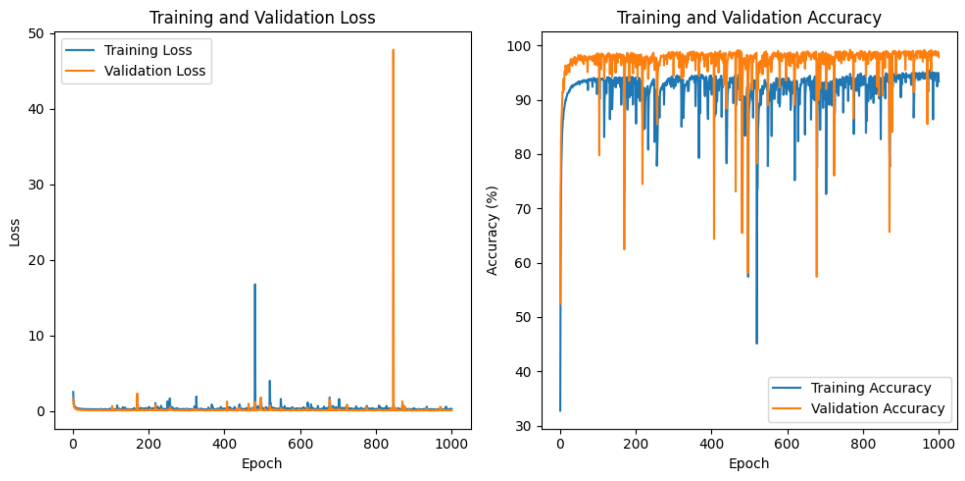
This comparison underscores DenseNet121's effectiveness in balancing performance and model size, and illustrates how transfer learning can drastically reduce computational de-mands while preserving much of the original model's accuracy.



(a)

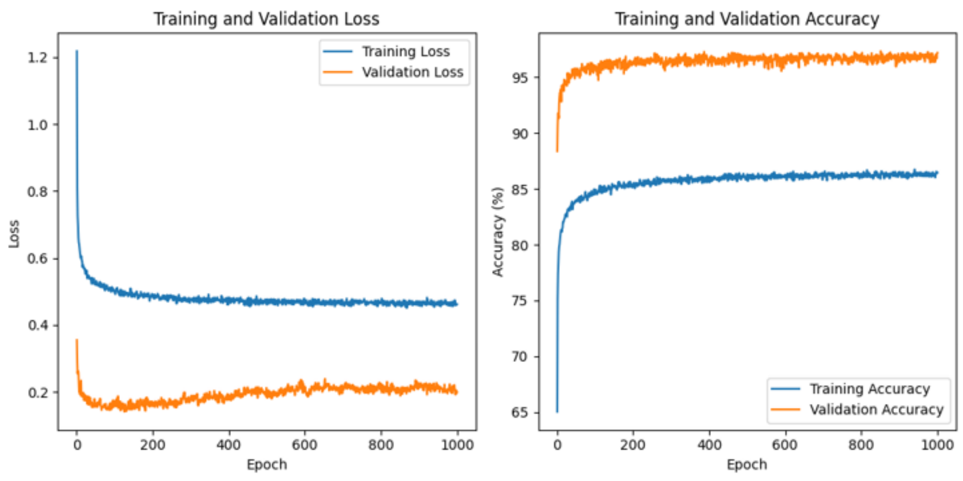


(b)

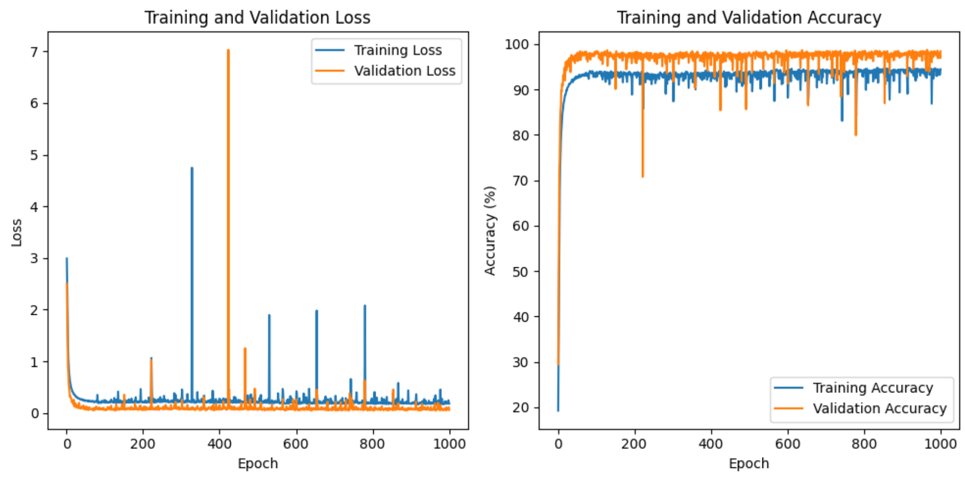


(c)

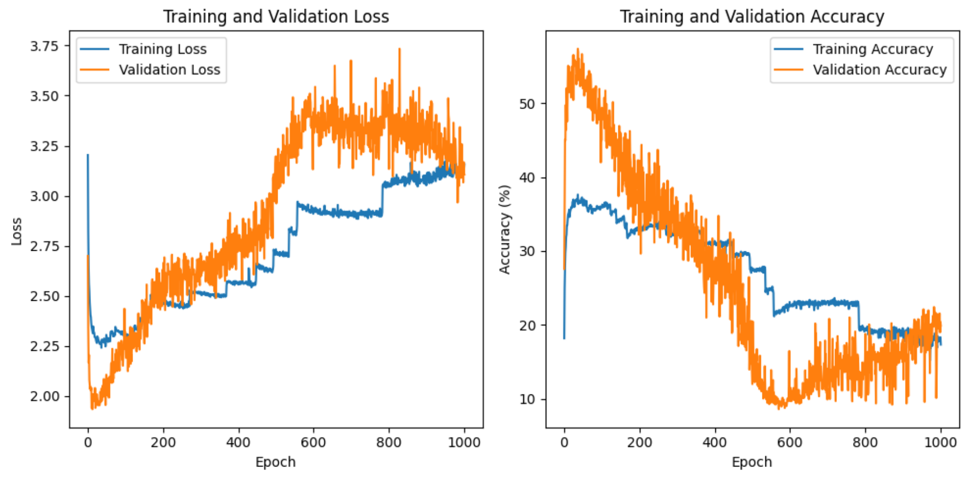
Fig 3.7: Accuracy and loss graphs on the PlantVillage database for: (a) DenseNet121, (b) AlexNet, (c) VGG19.



(a)



(b)



(c)

Fig 3.8: Accuracy and loss graph in PlantVillage database of : (a) TL DenseNet121, (b) TL AlexNet, (c) TL VGG 19.

3.10 Comparison with traditional methods

This section presents a comparative analysis of the proposed CNN models against traditional and recent deep learning approaches reported in the literature for plant disease detection. The comparison focuses on key performance metrics such as loss, accuracy, precision, recall, and F1 score to highlight improvements over previous methods.

The reviewed studies encompass a range of architectures, including MobileNet-V2 with attention mechanisms, Inception-based models enhanced by residual connections and channel-wise attention, Vision Transformers (ViT), and various CNN-based approaches such as VGG19 and ShuffleNet variants. Transfer learning adaptations of these models are also considered, reflecting current trends in leveraging pre-trained networks to improve performance and reduce training costs.

Our models, DeepCNN and SimpleCNN, developed in 2025, demonstrate superior or comparable performance relative to these prior methods, achieving accuracy and F1 scores exceeding 99% while maintaining a balanced trade-off between model complexity and computational efficiency. Notably, the SimpleCNN model attains high accuracy with a simpler architecture, suggesting practical advantages for deployment in resource-constrained environments.

This comparative evaluation underscores the ongoing progress in plant disease detection technologies and validates the effectiveness of our CNN-based approaches within this evolving landscape.

Author	**Year**	**Technique**	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Chen et al. [Chen et al., 2021b]	2021	MobileNet-V2, attention mechanism	0.017	96.68	97.49	95.83	96.64
Zhao et al. [Zhao et al., 2022]	2022	Inception, residual, modified channelwise attention module	0.12	97.28	97.49	97.06	97.27
Thakur et al. [Thakur et al., 2022b]	2022	PlantXViT	0.04	98.86	98.90	98.81	98.85
TABBAKH et al. [Tabbakh and Barpanda, 2023]	2023	VGG19 followed by ViT	–	98.81	98.72	98.76	98.73
Iftikhar et al. [Iftikhar et al., 2024]	2024	E-CNN Model	–	98.17	–	–	–
Bouacida et al. [PyTorch Team, 2024]	2024	DL-based system	–	94.04	95.05	93.95	95.50
Rizwan et al. [Rizwan et al., 2024]	2024	Proposed model using ShuffleNet	–	96.86	96.81	96.86	96.78
Shafik et al. [Shafik et al., 2024]	2024	PDDNet-LVE model	–	97.79	–	–	97.07
Ouamane et al. [Ouamane et al., 2025]	2024	optimized ViT model	0.009	99.77	99.61	99.67	99.64
Our	2025	DeepCNN	0.0177	99.53	99.37	99.30	99.33
Our	2025	SimpleCNN	0.0274	99.26	99.22	98.94	99.07

Table 3.6: Performance comparison across various plant disease detection research works.

When compared to models from prior years, the 2025 models (DeepCNN and SimpleCNN) exhibit notable improvement :

- Performance: Accuracy increased significantly from 96.68% in 2021 to 99.53% in 2025.
- Timeline progression:
  - 2021: 96.68% (Chen et al.)
  - 2022: 97.28-98.86% (Thakur et al., Zhao et al.)
  - 98.81% in 2023 (TABBAKH et al.)
  - 2024: 94.04-99.77% (different models)
  - 2025: according to our models, 99.26-99.53%
- Metric balance: The 2025 models exhibit superior reliability by maintaining an excellent balance between precision and recall (>99% for DeepCNN).
- Architectural efficiency: Compared to the intricate models from prior years, SimpleCNN (2025) achieves 99.26% accuracy with a simpler architecture.

Our 2025 models exhibit a remarkable optimisation of the complexity/performance trade-off, reflecting ongoing advancements in the field of plant disease detection.

3.11 Conclusion

The evaluation of both **DeepCNN** and **SimpleCNN** models demonstrated their high effectiveness in plant disease detection, achieving accuracy values of **99.53%** and **99.26%**, respectively. While SimpleCNN offers a computationally efficient architecture with only 25.8 million parameters, DeepCNN consistently outperformed it across all key performance metrics, including precision (99.37%) and loss (0.0177), while maintaining a reasonable inference time of 10.08 seconds. These results establish DeepCNN as the superior model in terms of robustness and predictive reliability.

Future research should focus on expanding dataset diversity to better capture real-world agricultural variability, optimizing model architectures for deployment on resource constrained devices, integrating Explainable AI techniques to improve model interpretability, and conducting extensive validations under diverse environmental and operational conditions to ensure practical applicability.

General Conclusion

This research makes a significant scientific contribution to the field of smart agriculture by addressing critical challenges in automated plant disease detection through advanced deep learning methodologies. Our findings demonstrate that integrating optimized convolutional neural network architectures with enhanced data preprocessing and augmentation strategies can achieve exceptionally high classification accuracy while maintaining computational efficiency suitable for real-world deployment.

Through rigorous evaluation across diverse datasets, our models established new performance benchmarks in plant disease classification, providing a reliable foundation for future research in this domain. The proposed standardized evaluation frameworks and comparative architectural analyses offer practical guidance for selecting context-appropriate models, thereby improving the reproducibility and scalability of AI-driven agricultural solutions.

The practical implications of this work are substantial. The optimized models are well-suited for integration into intelligent agricultural systems, including mobile diagnostic applications, drone-based field surveillance platforms, and IoT-enabled sensor networks. These applica-tions facilitate real-time monitoring and early disease detection, shifting the paradigm from reactive to proactive crop management, and thereby contributing to sustainable farming practices and enhanced global food security.

Despite these advancements, several limitations warrant further investigation. Existing da-tasets remain limited in their representation of the full spectrum of real-world agricultural variability. We emphasize the importance of developing comprehensive, geographically di-verse datasets through collaboration with farming communities. Additionally, ensuring mod-el generalization across diverse environmental conditions and reducing computational de-mands for deployment in resource-constrained settings are critical challenges for future re-search. Prioritizing lightweight architectures and real-time inference capabilities will be es-sential to enable large-scale deployment.

In conclusion, this study highlights the transformative potential of artificial intelligence in plant health management not only for accurate disease detection but also as a cornerstone for comprehensive crop optimization systems. The frameworks and findings presented here lay the groundwork for future innovations in intelligent agriculture, advancing the global mission of sustainable and efficient food production through scientifically grounded, AI-powered technologies.

Bibliography

[Con, 2020] (2020). Convolutional layer. https://commons.wikimedia.org/wiki/File:Conv_layer.png. Accessed: August 2020. (pages iv, 34).

[Max, 2020] (2020). Max pooling. https://commons.wikimedia.org/wiki/File:Max_pooling.png. Accessed: August 2020. (pages iv, 35).

[Typ, 2020] (2020). Typical CNN. https://commons.wikimedia.org/wiki/File:Typical_cnn.png. Accessed: August 2020. (pages iv, 32).

[icv, 2022] (2022). Icvt: Inception-based cnn with vision transformer for plant disease identification. *Computers and Electronics in Agriculture*, 197:106976. Accessed via ScienceDirect. (page 39).

[lit, 2024] (2024). Lite-mdc: A lightweight multi-kernel depthwise convolution network for plant disease recognition. *Smart Agricultural Technology*, 8:100590. Accessed via ScienceDirect. (page 39).

[Alzoubi et al., 2023] Alzoubi, S., Jawarneh, M., Bsoul, Q., Keshta, I., Soni, M., and Khan, M. A. (2023). An advanced approach for fig leaf disease detection and classification: Leveraging image processing and enhanced support vector machine methodology. *Open Life Sciences*, 18(1):20220764. (page 25).

[Arnal Barbedo, 2013] Arnal Barbedo, J. G. (2013). Digital image processing techniques for detecting, quantifying and classifying plant diseases. *SpringerPlus*, 2(1):660. (page 25).

[Barbedo, 2016] Barbedo, J. G. A. (2016). A review on the main challenges in automatic plant disease identification based on visible range images. *Biosystems engineering*, 144:52–60. (pages 19, 21).

[Chen et al., 2021a] Chen, J., Zhang, D., Suzauddola, M., and Zeb, A. (2021a). Identifying crop diseases using attention embedded mobilenet-v2 model. *Applied Soft Computing*, 113:107901. (page 2).

[Chen et al., 2021b] Chen, J., Zhang, D., Zeb, A., and Nanehkaran, Y. A. (2021b). Identification of rice plant diseases using lightweight attention networks. *Expert Systems with Applications*, 169:114514. (page 68).

[Chester, 1933] Chester, K. S. (1933). The problem of acquired physiological immunity in plants. *The Quarterly Review of Biology*, 8(3):275–324. (page 21).

[Conversation, 2020] Conversation, T. (2020). De nouvelles technologies pour mieux dÃ©tecter les maladies des plantes. Accessed on May 25, 2025. (pages 17, 18).

[Dordas, 2008] Dordas, C. (2008). Role of nutrients in controlling plant diseases in sustainable agriculture. a review. *Agronomy for sustainable development*, 28:33–46. (page 21).

[Dumoulin, 2020] Dumoulin, V. (2020). A comprehensive guide to convolutional neural networks: The eli5 way. Accessed: August 2020. (pages iv, 33).

[GeeksforGeeks, 2025] GeeksforGeeks (2025). Geeksforgeeks | your all-in-one learning portal. ConsultÃ© le 25 mai 2025. (pages 1, 57, 58).

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge. (pages 29, 38).

[Gupta et al., 2012] Gupta, D., Malviya, A., and Singh, S. (2012). Performance analysis of classification tree learning algorithms. *International Journal of Computer Applications*, 55(6). (page 59).

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. (page 38).

[Iftikhar et al., 2024] Iftikhar, M., Kandhro, I. A., Kausar, N., Kehar, A., Uddin, M., and Dandoush, A. (2024). Plant disease management: A fine-tuned enhanced cnn approach with mobile app integration for early detection and classification. *Artificial Intelligence Review*, 57(7):167. (page 68).

[Kamilaris and Prenafeta-Boldú, 2018] Kamilaris, A. and Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. *Computers and electronics in agriculture*, 147:70–90. (page 25).

[Khan et al., 2021] Khan, P., Kader, M. F., Islam, S. R., Rahman, A. B., Kamal, M. S., Toha, M. U., and Kwak, K.-S. (2021). Machine learning and deep learning approaches for brain disease diagnosis: principles and recent advances. *Ieee Access*, 9:37622–37655. (pages iv, 30).

[Khandagale and Others, 2025] Khandagale, Y. and Others (2025). Fourcropnet: A residual attention-based model for multi-crop plant disease detection. *Journal of Agricultural AI Research*. In press. (page 39).

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25. (pages 1, 38, 58).

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444. (pages 31, 38).

[LeCun et al., 1988] LeCun, Y., Haffner, P., Bottou, L., Bengio, Y., Bottou, L., Haffner, P., Howard, P., Simard, P., Bengio, Y., LeCun, Y., et al. (1988). Object recognition with gradient-based learning. *Feature Grouping*, 66:233–240. (page 32).

[LeCun et al., 2010] LeCun, Y., Kavukcuoglu, K., and Farabet, C. (2010). Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE international symposium on circuits and systems*, pages 253–256. IEEE. (pages 33, 34, 35, 37).

[Li et al., 2021] Li, L., Zhang, S., and Wang, B. (2021). Plant disease detection and classification by deep learning-a review. *IEEE Access*, 9:56683–56698. (page 27).

[Love The Garden, 2024] Love The Garden (2024). ReconnaÃ®tre les maladies vÃ©gÃ©tales les plus courantes. ConsultÃ© le 25 mai 2025. (pages 10, 11, 12).

[Love The Garden, 2025] Love The Garden (2025). Reconnaitre les maladies vegetales les plus courantes. Consulte le 23 mai 2025. (pages iv, 6, 7, 8, 9, 11, 12, 13).

[Malik, 2020] Malik, S. (2020). Understanding convolutional neural networks (cnn). Accessed: August 2020. (page 36).

[Masko and Hensman, 2015] Masko, D. and Hensman, P. (2015). The impact of imbalanced training data for convolutional neural networks. (page 32).

[Mohyuddin et al., 2024] Mohyuddin, G., Khan, M. A., Haseeb, A., Mahpara, S., Waseem, M., and Saleh, A. M. (2024). Evaluation of machine learning approaches for precision farming in smart agriculture system-a comprehensive review. *IEEE Access*. (page 2).

[Nair, 2023] Nair, K. P. (2023). *Biodiversity in Agriculture*. Springer. (page 2).

[Ouamane et al., 2025] Ouamane, A., Chouchane, A., Himeur, Y., Miniaoui, S., Atalla, S., Mansoor, W., and Al-Ahmad, H. (2025). Optimized vision transformers for superior plant disease detection. *IEEE Access*. (page 68).

[Palanisamy and Kandasamy, 2022] Palanisamy, P. and Kandasamy, R. (2022). 14-dcnn: A deep convolutional neural network for plant disease detection with optimized parameters. *Computers and Electronics in Agriculture*, 198:107075. (page 39).

[Paszke, 2019] Paszke, A. (2019). Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*. (page 45).

[Peng et al., 2021] Peng, J., Jury, E. C., Dönnes, P., and Ciurtin, C. (2021). Machine learning techniques for personalised medicine approaches in immune-mediated chronic inflammatory diseases: applications and challenges. *Frontiers in pharmacology*, 12:720694. (pages iv, 28, 29).

[PyTorch Team, 2024] PyTorch Team (2024). Pytorch: The open language of ai. ConsultÃ© le 25 mai 2025. (pages 44, 57, 68).

[Reis-Pereira et al., 2024] Reis-Pereira, M., Mazivila, S. J., Tavares, F., dos Santos, F. N., and Cunha, M. (2024). Early plant disease diagnosis through handheld uv-vis transmittance spectrometer with dd-simca one-class classification and mcr-als bilinear decomposition. *Smart Agricultural Technology*, 9:100631. (page 14).

[Rizvi, 2020] Rizvi, M. S. Z. (2020). Learn image classification using cnn: Convolutional neural networks with 3 datasets. Accessed: August 2020. (page 31).

[Rizwan et al., 2024] Rizwan, M., Bibi, S., Haq, S. U., Asif, M., Jan, T., and Zafar, M. H. (2024). Automatic plant disease detection using computationally efficient convolutional neural network. *Engineering Reports*, 6(12):e12944. (page 68).

[Salman et al., 2023] Salman, Z., Muhammad, A., Piran, M. J., and Han, D. (2023). Crop-saving with ai: latest trends in deep learning techniques for plant pathology. *Frontiers in Plant Science*, 14:1224709. (pages 24, 26).

[Shafik et al., 2024] Shafik, W., Tufail, A., De Silva Liyanage, C., and Apong, R. A. A. H. M. (2024). Using transfer learning-based plant disease classification and detection for sustainable agriculture. *BMC Plant Biology*, 24(1):136. (page 68).

[Shang et al., 2024] Shang, Z., Chauhan, V., Devi, K., and Patil, S. (2024). Artificial intelligence, the digital surgeon: Unravelling its emerging footprint in healthcare–the narrative review. *Journal of Multidisciplinary Healthcare*, pages 4011–4022. (pages iv, 30).

[Shoaib et al., 2023] Shoaib, M., Shah, B., Ei-Sappagh, S., Ali, A., Ullah, A., Alenezi, F., Gechev, T., Hussain, T., and Ali, F. (2023). An advanced deep learning models-based plant disease detection: A review of recent research. *Frontiers in Plant Science*, 14:1158933. (page 26).

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. (page 38).

[Sladojevic et al., 2016] Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., and Stefanovic, D. (2016). Deep neural networks based recognition of plant diseases by leaf image classification. *Computational intelligence and neuroscience*, 2016(1):3289801. (pages iv, 32).

[Streets, 1972] Streets, R. B. (1972). *The Diagnosis of Plant Diseases: A Field and Laboratory Manual Emphasizing the Most Practical Methods for Rapid Identification*. University of Arizona Press, Tucson, AZ. (page 20).

[SuperDataScience Team, 2020] SuperDataScience Team (2020). Convolutional neural networks: Step 3: Flattening. Accessed: August 2020. (pages 1, 36).

[Tabbakh and Barpanda, 2023] Tabbakh, A. and Barpanda, S. S. (2023). A deep features extraction model based on the transfer learning model and vision transformer "tlmvit" for plant disease classification. *IEEE Access*, 11:45377–45392. (page 68).

[Thakur et al., 2022a] Thakur, A., Sharma, D., and Singh, M. (2022a). Plantxvit: Explainable vision transformer for plant disease classification with lime and grad-cam. *Expert Systems with Applications*, 200:116903. (page 39).

[Thakur et al., 2022b] Thakur, P. S., Khanna, P., Sheorey, T., and Ojha, A. (2022b). Explainable vision transformer enabled convolutional neural network for plant disease identification: Plantxvit. *arXiv preprint arXiv:2207.07919*. (page 68).

[Wikipedia contributors, 2020] Wikipedia contributors (2020). RÃ©seau neuronal convolutif. Accessed: August 2020. (pages 34, 37).

[Yağ and Altan, 2022] Yağ, İ. and Altan, A. (2022). Artificial intelligence-based robust hybrid algorithm design and implementation for real-time detection of plant diseases in agricultural environments. *Biology*, 11(12):1732. (page 25).

[Yao et al., 2023] Yao, J., Tran, S. N., Sawyer, S., and Garg, S. (2023). Machine learning for leaf disease classification: data, techniques and applications. *Artificial Intelligence Review*, 56(Suppl 3):3571–3616. (page 24).

[Yasin and Fatima, 2023] Yasin, A. and Fatima, R. (2023). On the image-based detection of tomato and corn leaves diseases: An in-depth comparative experiments. *arXiv preprint arXiv:2312.08659*. (page 39).

[Zhang et al., 2020] Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. (2020). Dive into deep learning: The convolutional layer. Accessed: August 2020. (pages iv, 34).

[Zhao et al., 2022] Zhao, Y., Sun, C., Xu, X., and Chen, J. (2022). Ric-net: A plant disease classification model based on the fusion of inception and residual structure and embedded attention mechanism. *computers and Electronics in Agriculture*, 193:106644. (page 68).