Efficient Plant Disease Detection Using Hybrid Deep Learning Models

by

Emmanuel O. Udoh

A dissertation submitted to the
Department of Computer Science
in conformity with the requirements for
the degree of Master of Science

Bishop's University
Canada
May 2026

Abstract

Plant diseases cause substantial losses in agricultural productivity, and traditional identification methods relying on visual inspection by experts are often slow, subjective, and impractical for large-scale monitoring. Conventional deep learning models, while effective, tend to be computationally expensive and unsuitable for deployment in resource-constrained environments such as mobile devices. To address these challenges, this thesis investigates efficient hybrid deep learning models for plant disease detection, with a particular focus on enhancing MobileNetV2, a lightweight architecture that employs depthwise separable convolutions to drastically reduce computational cost while maintaining high representational power. The proposed approach integrates a Convolutional Block Attention Module (CBAM) with MobileNetV2 to improve feature selection through channel and spatial attention without significantly increasing model complexity. Experiments were conducted on the PlantVillage dataset comprising approximately 54,000 images across 38 disease and healthy leaf classes. The hybrid MobileNetV2+CBAM model was compared against baseline MobileNetV2, Inception, and Xception networks, with training aided by data augmentation, regularization, and adaptive learning rate scheduling. Results demonstrate that the CBAM-enhanced MobileNetV2 achieves improved classification accuracy over the baseline MobileNetV2 model while preserving computational efficiency. However, since the evaluation is conducted on the PlantVillage dataset, which consists of images captured under controlled conditions, the reported performance should be interpreted within the context of benchmark evaluation. Further validation on real-world field data is required to fully assess generalization and deployment readiness in practical agricultural environments.

Acknowledgments

First and foremost, I express my deepest gratitude to my supervisors, Dr. Madjid Allili and Dr. Mohammed Ayoub Alaoui Mhamdi for their invaluable guidance, expertise, thoughtful feedback, constructive criticism, and continuous support throughout the course of this research.

Special appreciation is extended to our graduate coordinator, Dr. Stefan Bruda, whose unwavering administrative and academic support ensured a smooth and productive research journey. I would also like to sincerely thank the chair of my examining committee, Dr. Yasir Malik, along with the committee members, Dr. Russell Butler and Dr. Rachid Hedjam, for their careful evaluation of this work and for the numerous insightful comments and suggestions that significantly contributed to improving the quality and clarity of this thesis. Their feedback was instrumental in strengthening this research.

Finally, a heartfelt gratitude to my beloved mother Mrs. Tosin Atinuke Ajose-Udoh, whose emotional support and encouragement sustained me through the most challenging periods of this endeavor. This journey would not have been possible without the collective contributions of these remarkable individuals. I am sincerely thankful.

Contents

1 Introduction **1**
- 1.1 Problem Statement . . . 2
- 1.2 Research Objectives . . . 2

2 Literature Review **5**
- 2.1 Background . . . 5
- 2.2 Plant Disease Detection . . . 6
- 2.3 Traditional Methods of Disease Detection . . . 7
- 2.4 Technological Approaches to Plant Disease Detection . . . 8
- 2.5 Basic Steps in Plant Disease Detection . . . 9
- 2.6 Related Work . . . 11
- 2.7 Deep Learning Models for Disease Classification . . . 13
- 2.8 Hybrid Model Classifiers . . . 16
- 2.9 Multi-Disease Classification and Transfer Learning . . . 17
- 2.10 Depthwise Separable Convolutions . . . 18
- 2.11 Attention Mechanism . . . 19
- 2.12 Convolutional Block Attention Module (CBAM) . . . 21
- 2.13 IoT and Edge Computing Integration . . . 23
- 2.14 Spectral & Hyperspectral Imaging for Disease Detection . . . 23
- 2.15 Responsible AI (RAI) and Explainable AI (XAI) in Plant Disease Detection . . . 24

3 Methodology **26**
- 3.1 Overview of Methodology . . . 26
- 3.2 Dataset Description . . . 27
- 3.3 Data Preprocessing . . . 29
- 3.4 Model Architecture . . . 30
- 3.5 Training Configuration . . . 32
- 3.6 Evaluation Metrics . . . 33
- 3.7 Visualization Tools . . . 35
- 3.8 Deployment Pipeline . . . 37
- 3.9 Summary . . . 38

4 Experimental Results and Analysis **40**
4.1 Experimental Setup 40
4.2 Performance Metrics Summary 41
4.3 Comparative Model Evaluation 44
4.4 Model Per-Class Classification Evaluation 48
4.5 Grad-CAM Visualizations 57
4.6 Statistical Discussion & Observations 59
4.7 Deployment Performance on Mobile 60

5 Conclusion and Future Work **62**
5.1 Conclusion 62
5.2 Research Contributions 63
5.3 Future Work 63

Bibliography **64**

List of Tables

3.1 Sample distribution of selected classes from the PlantVillage dataset 27
3.2 Comparison of model parameters and size 32

4.1 Model performance comparison on test set 41
4.2 Experimental results reported by Mohanty et al. 42
4.3 Comparison of classification accuracy with Mohanty et al. 43
4.4 Training duration of models (30 Epochs) 43
4.5 Per-class classification performance of MobileNetV2 (RHOMBUS) . . 51
4.6 Per-class classification performance of MobileNetV2 + CBAM (CUBE) 53
4.7 Per-class classification performance of Xception (PRISM) 54
4.8 Per-class classification performance of Inception (CONE) 56
4.9 Model performance metrics on test set 59
4.10 Quantized model deployment metrics on mobile device 60

List of Figures

2.1 The CBAM architecture. 22

3.1 Sample images from different classes in the PlantVillage dataset. . . . 28
3.2 Sample confusion matrix for the proposed model. 34
3.3 Example ROC curves for selected classes. 36
3.4 Per-class classification accuracy for the proposed model. 37

4.1 Comparison of test accuracy across all models 44
4.2 Comparison of test F1-score across all models 44
4.3 Comparison of test AUC score across all models 45
4.4 RHOMBUS (MobileNetV2): Training vs validation 46
4.5 PRISM (Xception): Training vs validation 47
4.6 CONE (InceptionV3): Training vs validation 49
4.7 CUBE (MobileNetV2 + CBAM): Training vs validation 50
4.8 Grad-CAM visualization comparison for example leaf images across RHOMBUS, PRISM, CONE, and CUBE 58

Chapter 1

Introduction

Agricultural productivity is under increasing threat due to plant diseases, which account for up to 40% of crop losses globally each year, translating to over US $220 billion in economic damage and food security risks—it is a compelling global challenge affecting rural incomes and livelihoods [17, 43]. For instance, wheat, rice, maize, and potatoes can individually sustain yield losses exceeding 20–30% when afflicted by common pathogens [9]. Traditional diagnostic methods rely heavily on expert visual inspection or laboratory tests, which are time-consuming, subjective, and impractical for widespread or real-time field deployment.

In contrast, deep learning, particularly convolutional neural networks (CNNs), has shown remarkable success in image-based plant disease classification. Early work by Mohanty et al. utilized CNNs on the PlantVillage dataset (over 50,000 images across multiple disease classes) to achieve >99% accuracy under controlled conditions [28]. However, such models are computationally heavy and often unsuitable for mobile deployment due to large parameter counts and high latency.

Recent advances in efficient architectures (e.g., MobileNet) and lightweight attention mechanisms (e.g., CBAM) provide a promising path forward. MobileNet, built on depthwise separable convolutions, reduces computation while maintaining strong representation capacity [4]. Meanwhile, CBAM (Convolutional Block Attention Module) refines feature learning via channel and spatial attention with minimal overhead [44]. By combining these techniques, it is possible to develop models that are both accurate and deployable on low-power devices.

Therefore, the motivation of this research is clear: to design and validate a hybrid deep learning model that bridges the gap between high-accuracy disease detection and practical field deployment. Such a solution could empower farmers with real-time, AI-driven diagnostics, enabling timely intervention and improving crop health monitoring in resource-limited settings.

1.1 Problem Statement

Plant diseases pose a significant threat to global agricultural productivity and food security, causing substantial economic losses and reduced crop yields each year. Early and accurate identification of plant diseases is therefore essential for effective crop management and sustainable agricultural practices. Traditionally, disease detection relies on manual inspection by agricultural experts, a process that is often time-consuming, subjective, and impractical for large-scale monitoring in modern farming environments [17]. As a result, automated image-based disease detection systems have gained increasing attention in recent years.

Advancements in deep learning, particularly Convolutional Neural Networks (CNNs), have shown promising results in plant disease classification tasks using leaf images. Several studies have demonstrated high classification accuracy using deep architectures such as AlexNet, GoogLeNet, and ResNet on datasets like PlantVillage [28]. However, many of these models are computationally intensive and require significant processing resources, which limits their practical deployment on mobile or edge devices commonly used in real-world agricultural settings.

Furthermore, conventional CNN architectures may struggle to effectively capture subtle disease symptoms, especially when lesions or discolorations occupy small regions of leaf images. Recent research suggests that integrating attention mechanisms into deep learning architectures can enhance feature representation by enabling models to focus on the most relevant spatial and channel-wise information [44]. Despite these developments, there remains a need for an efficient hybrid deep learning framework that combines lightweight architectures with attention mechanisms to improve both classification accuracy and computational efficiency for practical plant disease detection applications.

1.2 Research Objectives

1.2.1 General Objective

The main objective of this research is to develop an efficient hybrid deep learning framework for accurate plant disease detection from leaf images by integrating lightweight convolutional neural network architectures with attention mechanisms to improve classification performance while maintaining computational efficiency.

1.2.2 Specific Objectives

To achieve the overall goal of this study, the following specific objectives are defined:

- Build a high-performance deep learning system for identifying 38 different plant diseases.

- To investigate existing deep learning approaches used for image-based plant disease detection and compare multiple CNN-based models: MobileNetV2, Xception, Inception, and a custom attention-enhanced variant.
- To propose a hybrid deep learning model that balances accuracy and efficiency.
- To ensure the model can be deployed on mobile devices for real-time, field-level diagnosis.

1.2.3 Research Questions

- **RQ1 (Efficiency–Accuracy Trade-off):**

  How can we minimize computational complexity (parameters, FLOPs, model size, latency) without sacrificing accuracy on PlantVillage? What Pareto frontier can be achieved across candidate backbones (MobileNetV2, Xception, Inception) and the proposed MobileNetV2+CBAM hybrid?

- **RQ2 (Attention Benefits):**

  To what extent does integrating CBAM into a lightweight backbone (MobileNetV2) improve class-wise precision/recall/F1 and AUC versus the same backbone without attention and versus heavier baselines?

- **RQ3 (Training Strategy):**

  Which fine-tuning choices (layers unfrozen, learning-rate schedules, dropout/L2 regularization) most effectively boost generalization while controlling overfitting in a low-compute regime?

- **RQ4 (Method Comparison):**

  How does the proposed hybrid deep learning framework compare with existing plant disease detection models in terms of accuracy and efficiency?

- **RQ5 (Deployability):**

  Can the proposed model provide a practical solution for automated plant disease detection suitable for deployment in real-world agricultural environments?

1.2.4 Significance of the Study

This study contributes to the advancement of intelligent agricultural systems by developing an efficient deep learning-based framework for automated plant disease detection from leaf images. By integrating lightweight convolutional neural network architectures with attention mechanisms, the proposed approach aims to

improve the accuracy and efficiency of disease classification while reducing computational complexity. Such improvements are particularly important for enabling practical deployment on mobile or resource-constrained devices used by farmers and agricultural practitioners. The outcomes of this research can support early disease diagnosis, reduce crop losses, and enhance decision-making in crop management. Furthermore, the study contributes to the growing body of research in precision agriculture by demonstrating how hybrid deep learning models can be leveraged to build scalable and accessible solutions for real-world agricultural monitoring and plant health management.

1.2.5 Scope and Limitations of the Study

This study focuses on the development and evaluation of a hybrid deep learning framework for automated plant disease detection using leaf images. The research utilizes the PlantVillage Dataset, which contains labeled images of healthy and diseased plant leaves across multiple crop species. The proposed approach investigates the performance of pretrained convolutional neural network architectures, specifically MobileNetV2 and Xception, enhanced with the Convolutional Block Attention Module (CBAM) to improve feature representation and classification accuracy. The study primarily addresses the classification of plant diseases from static leaf images and evaluates model performance using standard metrics such as accuracy, precision, recall, and F1-score.

However, several limitations should be acknowledged. First, the study relies on images from the PlantVillage dataset, which are captured under controlled conditions and may not fully represent the variability present in real-world agricultural environments, such as varying lighting conditions, complex backgrounds, and occlusions. Second, the research focuses on image-based disease detection and does not incorporate other potential data sources such as environmental factors, soil conditions, or temporal disease progression. Additionally, while the study aims to develop computationally efficient models suitable for mobile deployment, full real-world implementation and field validation are beyond the scope of this research. Despite these limitations, the findings provide valuable insights into the application of hybrid deep learning models for efficient plant disease detection in precision agriculture.

Chapter 2

Literature Review

2.1 Background

Plant pathology or phytopathology is the scientific discipline concerned with the study of plant diseases, their causal agents, mechanisms of infection, and strategies for control. Plant diseases, broadly defined as abnormal physiological or morphological conditions that negatively affect plant growth and yield, are caused by a wide range of biotic (living) and abiotic (non-living) factors [1]. Understanding the nature and categories of these diseases is essential for developing effective detection and management strategies.

2.1.1 Biotic Diseases

Biotic diseases are caused by infectious organisms such as fungi, bacteria, viruses, and nematodes.

- **Fungal diseases** are caused by fungi, which are eukaryotic organisms that live on or within plants and obtain nutrients from the host. They account for the majority of plant infections, with examples including powdery mildew, late blight (caused by Phytophthora infestans), and rust diseases. These pathogens spread rapidly in humid environments and can devastate staple crops like potatoes, wheat, and maize [9].
- **Bacterial diseases** are caused by bacteria, single-celled prokaryotic organisms that can rapidly multiply within plant tissues, such as bacterial wilt (Ralstonia solanacearum) and citrus canker (Xanthomonas axonopodis), often cause systemic infection, leading to vascular blockage and wilting [13].
- **Viral diseases** are caused by viruses, which are microscopic infectious agents that replicate inside host plant cells. Examples include cucumber mosaic virus, tomato spotted wilt virus and banana bunchy top virus, are typically

transmitted by insect vectors, making them difficult to control. Their symptoms include mosaic patterns, leaf curling, and severe yield reduction [18].

- **Nematode diseases** arise from microscopic roundworms (nematodes) that attack plant roots, disrupting nutrient uptake and hindering growth. For example, root-knot nematodes can significantly reduce productivity in vegetables [31].

2.1.2 Abiotic Diseases

Abiotic plant disorders arise from environmental or chemical stressors rather than living organisms. These include nutrient deficiencies, soil salinity, drought, frost, and exposure to pollutants. For instance, iron deficiency chlorosis results in yellowing leaves due to impaired chlorophyll synthesis, while ozone toxicity can cause necrotic spotting [26]. Although not infectious, abiotic stresses often predispose plants to secondary infections by biotic agents.

2.2 Plant Disease Detection

Plant disease detection is the process of recognizing and classifying diseases affecting plants, encompassing a wide range of pathogens including fungi, bacteria, viruses, and nematodes. Accurate and timely identification is crucial for implementing appropriate management strategies to minimize yield losses, prevent widespread epidemics, and ensure food security, impacting both the quantity and quality of food production.

2.2.1 Importance of Plant Disease Detection

Accurate and timely identification of plant diseases is critical for ensuring global food security and sustaining agricultural productivity. Early detection allows farmers to initiate appropriate control measures, thereby reducing crop losses, minimizing the use of pesticides, and maintaining the quality of agricultural products [11]. Since approximately 80% of the world's food supply is derived from plants, their vulnerability to diseases directly influences human nutrition and economic stability [41].

Plant disease identification also serves as a foundation for precision agriculture, where advanced technologies enable targeted interventions. By detecting disease outbreaks at an early stage, farmers can optimize resource usage such as water, fertilizers, and pesticides, reducing environmental impacts while improving yields [12]. Thus, efficient disease identification is not only a biological necessity but also a technological enabler of sustainable farming systems.

2.2.2 Challenges in Plant Disease Detection

Despite its importance, plant disease identification remains a challenging task. Traditional methods rely on visual inspection by experts, which is subjective, time-consuming, and often limited by overlapping symptoms among different diseases [2]. For example, leaf yellowing may indicate viral infection, nutrient deficiency, or environmental stress, complicating accurate diagnosis. Furthermore, many pathogens exhibit high variability in symptom presentation depending on environmental conditions, plant age, or host resistance mechanisms [11].

The scarcity of trained pathologists in many developing regions exacerbates the challenge. Additionally, laboratory-based molecular methods such as polymerase chain reaction (PCR) provide higher accuracy but are resource-intensive and impractical for large-scale field applications [14]. These limitations highlight the pressing need for automated, scalable solutions that leverage computer vision and deep learning to overcome human subjectivity and enhance accuracy under real-world conditions.

2.2.3 Economic Impact of Disease Detection on Agriculture

The economic implications of plant diseases are immense, with global annual crop losses estimated at 20–40% due to pests and pathogens [7]. For instance, rice blast disease alone is responsible for destroying enough rice to feed 60 million people each year [10]. Similarly, late blight in potatoes caused by Phytophthora infestans remains one of the most economically damaging plant diseases in history, with devastating consequences during both historical events (e.g., the Irish famine) and modern outbreaks [5].

Accurate detection and management of plant diseases can substantially reduce these losses and improve food security. Early and precise identification prevents unnecessary pesticide use, lowering costs for farmers and mitigating environmental harm. Moreover, the integration of deep learning-based detection systems has the potential to democratize access to diagnostic expertise, allowing smallholder farmers—who produce a significant share of global food—to mitigate crop loss and increase profitability [28]. Thus, investment in efficient plant disease detection technologies is both an economic imperative and a pathway toward global agricultural resilience.

2.3 Traditional Methods of Disease Detection

Traditional plant disease detection methods have historically relied on visual inspection by farmers and experts, which, although inexpensive and straightforward, is often subjective, time-consuming, and prone to misdiagnosis, particularly when diseases present overlapping symptoms [25]. To improve accuracy,

laboratory-based approaches such as microscopy and culturing, serological assays like enzyme-linked immunosorbent assay (ELISA), and molecular diagnostics such as polymerase chain reaction (PCR) have been widely adopted [13, 14]. While these methods provide reliable identification, they are labor-intensive, costly, and unsuitable for large-scale or real-time field applications, especially in regions with limited access to laboratories and trained specialists. Consequently, the limitations of traditional approaches highlight the urgent need for automated, efficient, and scalable solutions, such as deep learning–based image recognition, which can deliver timely and accurate plant disease detection directly in the field [28].

2.4 Technological Approaches to Plant Disease Detection

The adoption of digital imaging has enabled researchers to apply image processing and computer vision techniques for plant disease detection. Early approaches involved extracting handcrafted features such as color histograms, texture descriptors, and shape-based metrics to distinguish healthy and diseased tissues [2]. Classical algorithms including k-means clustering, support vector machines (SVM), and random forests were then employed to classify these features into disease categories [8]. Although such methods improved upon manual inspection, their performance often degraded under real field conditions due to variations in lighting, background clutter, and leaf morphology. Moreover, handcrafted features lacked the ability to generalize across diverse crops and complex disease symptoms, which limited their scalability.

2.4.1 The Role of Artificial Intelligence in Agriculture

The broader integration of artificial intelligence (AI) in agriculture has transformed multiple aspects of the food production pipeline, including crop monitoring, yield prediction, weed detection, and precision irrigation [21]. Machine learning algorithms and sensor-based technologies have enabled farmers to optimize resource allocation, reduce chemical usage, and improve sustainability. For plant disease management specifically, AI systems can analyze multispectral imagery, weather data, and field sensor readings to predict outbreaks before they become visually detectable [23]. This predictive capability highlights AI's role not only in diagnosis but also in prevention, supporting sustainable practices in line with global food security goals.

2.4.2 How AI Enhances Disease Identification

AI algorithms can learn from large datasets of plant images and spectral data, identifying complex relationships between visual and spectral features and disease presence, often surpassing human capabilities in detecting subtle patterns. This

allows for more accurate and objective disease classification compared to traditional methods, reducing reliance on subjective human judgement. AI can also be used to predict disease outbreaks based on environmental factors and historical data, enabling preventative measures to be taken before diseases spread widely. Furthermore, AI can be integrated with other technologies, such as robotics and drones, for automated disease scouting and targeted application of treatments [35]. The continuous development of new AI algorithms and increasing computational power are driving further advancements in this field.

2.5 Basic Steps in Plant Disease Detection

The process of plant disease detection typically follows a structured pipeline consisting of image acquisition, preprocessing, segmentation, feature extraction, and classification. Each step plays a crucial role in ensuring accurate and robust disease identification.

2.5.1 Image Acquisition

Image acquisition is the foundation of any automated disease detection system. It involves capturing leaf images using digital cameras, smartphones, or specialized imaging sensors such as multispectral or hyperspectral devices [23]. The quality of acquired images significantly influences subsequent processing steps; poor resolution, inconsistent lighting, or motion blur can degrade model performance. In field-based applications, acquisition methods must account for environmental variability such as changing light conditions and complex backgrounds.

2.5.2 Preprocessing and Augmentation

Preprocessing improves the quality and consistency of input images. Common techniques include resizing, color normalization, noise reduction, and contrast enhancement [39]. To improve model generalization, data augmentation is applied by artificially expanding the dataset through rescaling, rotation, translation, shearing, flipping, zooming, and brightness adjustments, simulating real-world variability. These steps reduce the risk of overfitting, particularly in deep learning models trained on limited or imbalanced datasets. Preprocessing aims to normalize nuisance factors: color constancy/white balancing, denoising, contrast normalization (e.g., CLAHE), and geometric normalization (resize/aspect policy). Data augmentation increases effective sample size and reduces overfitting via random flips/rotations, crops, scale and brightness/contrast jitter, cutout/mixup, and class-balanced sampling. Well-designed augmentation reflects plausible field variability and is particularly beneficial with limited labeled data.

2.5.3 Image Segmentation

Image segmentation is a computer vision technique that divides an image into multiple distinct regions or segments based on specific characteristics such as color, texture, or shape. Image segmentation aims to simplify an image's representation, making it easier to analyze and extract valuable information from different parts of the image. Technically, segmentation is how we can identify objects, people, or other important elements in the image using pixels [34]. More advanced methods leverage deep learning-based segmentation models such as U-Net or YOLO for precise localization of lesions. Accurate segmentation ensures that irrelevant background features do not interfere with disease classification, thereby enhancing diagnostic reliability.

Image segmentation algorithms use various mathematical models, machine learning algorithms, or deep neural networks to classify pixels or groups of pixels within an image. These algorithms analyze the visual features of the image and assign each pixel to a specific segment or region. Several high-level image features can be useful for image segmentation. Image segmentation can be categorized into several types, including:

- **Thresholding Segmentation:** This technique involves setting a threshold value to separate pixels based on their intensity or color.
- **Edge-based Segmentation:** It identifies object boundaries by detecting edges or gradients within the image.
- **Region-based Segmentation:** This approach groups pixels based on their visual features, color, or texture similarity.
- **Semantic segmentation:** Semantic segmentation is the process of assigning a class label to each pixel in the image so that pixels with the same label belong to the same object or category.
- **Clustering-based Segmentation:** It utilizes clustering algorithms to group pixels with similar attributes.
- **Instance segmentation** is the process of assigning a class label and an instance label to each pixel in the image, such that pixels with the same class label and instance label belong to the same object or instance.

2.5.4 Feature Extraction

Feature extraction translates visual patterns into numerical descriptors for classification. In traditional computer vision, handcrafted features such as color histograms, texture descriptors (GLCM, LBP), and shape attributes were commonly used. In

deep learning, convolutional neural networks (CNNs) automatically learn hierarchical feature representations directly from pixel values, eliminating the need for manual design. This step is central to distinguishing subtle disease patterns, such as variations in lesion color or texture.

2.5.5 Disease Classification

The final step involves classifying the extracted features into disease categories. Conventional methods employed classifiers like support vector machines (SVMs), decision trees, or k-nearest neighbors (KNN) [15]. In contrast, modern approaches rely on CNNs and their variants (e.g., ResNet, MobileNet, Xception) for end-to-end learning. Hybrid models, such as those incorporating attention mechanisms like CBAM, further enhance classification accuracy by focusing on the most informative image regions. Performance is typically evaluated using metrics such as accuracy, precision, recall, F1-score, and AUC.

2.6 Related Work

The application of deep learning to plant disease detection has received significant attention over the past decade, with researchers leveraging convolutional neural networks (CNNs) to achieve high classification accuracy using leaf images. One of the foundational works in this domain is by Mohanty, Hughes, and Salathé (2016), who utilized AlexNet and GoogLeNet architectures on the PlantVillage dataset—comprising over 54,000 labeled images across 38 classes—and reported near-perfect accuracy under controlled conditions [28]. While this established a strong benchmark, the study highlighted limitations regarding real-world deployment, where environmental variability often causes models to underperform.

Sladojević et al. (2016) similarly demonstrated an end-to-end CNN pipeline capable of automatically identifying diseases in plant leaves with minimal preprocessing [15]. Their model achieved high accuracy on curated datasets and emphasized the practical utility of CNNs in agricultural diagnostics. However, like Mohanty et al., the architecture used was relatively heavy, making it unsuitable for deployment on mobile or edge devices.

To benchmark the performance of various CNN architectures in the agricultural context, Ferentinos (2018) conducted an exhaustive comparison of models such as VGG, AlexNet, and ResNet on multiple crop types and disease categories [16]. The study confirmed the high performance of deep CNNs but also noted their computational demands, thus underscoring the need for lightweight, efficient alternatives that could retain accuracy while reducing resource consumption.

Addressing the need for real-world robustness, Singh et al. (2020) introduced the PlantDoc dataset, which includes in-the-wild images of plant diseases with varied lighting, occlusions, and complex backgrounds [40]. Their findings revealed

that models trained on clean datasets like PlantVillage suffer significant accuracy drops when evaluated on field data, emphasizing the necessity for architectures that generalize well across domains.

To bridge the performance gap between research and application, Ramcharan et al. (2019) developed and field-tested a cassava disease diagnosis app powered by CNNs and deployed on Android smartphones [32]. Their work validated the feasibility of on-device inference using lightweight models, while also revealing latency and memory limitations that need to be considered for mobile deployment in resource-constrained environments.

Recent research has focused on enhancing model performance without increasing computational burden. Woo et al. (2018) proposed the Convolutional Block Attention Module (CBAM), a lightweight plug-in module that sequentially applies channel and spatial attention to refine intermediate CNN features [44]. CBAM has since been widely adopted to boost model accuracy in image classification tasks, including plant disease recognition, with minimal additional cost. Similarly, Tan and Le (2019) introduced the EfficientNet family of models, which use compound scaling to balance network depth, width, and resolution for optimal accuracy-efficiency trade-off [42].

For mobile deployment, Sandler et al. (2018) developed MobileNetV2, incorporating inverted residual blocks and depthwise separable convolutions to drastically reduce the number of parameters and computational operations while maintaining competitive accuracy [36]. This architecture has become a standard for real-time applications and is a central backbone in the development of hybrid models such as MobileNetV2+CBAM.

To improve model interpretability, Selvaraju et al. (2017) proposed Grad-CAM, a technique that generates class-discriminative heatmaps for CNN predictions [38]. Grad-CAM is now widely used in plant disease studies to verify whether models focus on symptomatic regions of the leaf, thereby increasing transparency and trust in AI-driven agricultural tools.

Finally, Chollet (2017) introduced Xception, a deep CNN architecture built entirely from depthwise separable convolutions with residual connections [3]. It is often used as a strong baseline in transfer learning and plant disease classification due to its balance between depth and computational efficiency.

In summary, the literature reveals a trend toward optimizing the trade-off between accuracy and computational complexity. Lightweight architectures like MobileNetV2, when enhanced with modules such as CBAM, present a promising path for building efficient yet high-performing models suitable for real-time plant disease detection and mobile deployment. These hybrid designs, along with tools for visual interpretability like Grad-CAM, form the conceptual and technical foundation of this thesis.

2.7 Deep Learning Models for Disease Classification

Deep learning models, particularly Convolutional Neural Networks (CNNs), have revolutionized disease classification by automatically learning hierarchical feature representations from raw data [27]. These models can detect subtle patterns in medical images (like X-rays, MRIs, histopathology slides) or plant images that correlate with diseases, often achieving expert-level accuracy in tasks such as disease detection, localization, and prognosis. Below, we introduce key architectures, highlighting their history, design, equations, and relevance for disease classification.

2.7.1 Convolutional Neural Networks (CNN)

CNNs are a class of deep neural networks specialized for grid-structured data like images. *Convolutional layers* within a CNN apply learnable filters (kernels) that slide over the input's spatial dimensions, performing dot-products to produce feature maps. This convolution operation can be expressed as follows. Let x be the input image (or feature map) of size $\mathbf{H} \times \mathbf{W} \times \mathbf{C}_{in}$ and a filter $\mathbf{W}$ of size $\mathbf{K}_h \times \mathbf{K}_w$ applied to produce output channel k. The convolution output y_k at spatial position (i, j) is:

$$y_k(i,j) = \sum_{c=1}^{C_{in}} \sum_{u=1}^{K_h} \sum_{v=1}^{K_w} x_c(i+u-1, j+v-1) W_{c,k}(u,v) + b_k \tag{2.1}$$

where c indexes input channels, u,v index the filter's height and width, and b_k is the bias term. Each output feature $y_k(i, j)$ is the weighted sum of a local $\mathbf{K}_h \times \mathbf{K}_w$ patch of input pixels across all channels, using the filter's weights $\mathbf{W}_{c,k}$.

Pooling layer reduces the spatial dimensions of feature maps, decreasing computational complexity and making the network more robust to spatial variations in the input [29]. Max pooling, the most widely used pooling operation, selects the maximum value from a region of the feature map, as defined by:

$$P_{i,j} = \max_{a,b \in N_{ij}} \mathbf{X}_{ab} \tag{2.2}$$

where P_{ij} is the output at position (i, j), $\mathbf{X}_{ab}$ represents the element in the pooling window, and N_{ij} is the neighbourhood covered by the pooling kernel.

Fully connected (FC) layers integrate features learned by previous layers into the final prediction or classification. Each neuron in the FC layer connects to all activations from the preceding layer, forming a dense network that learns complex relationships among features. The output of an FC layer is computed as

$$\mathbf{y} = \sigma(W_x + b) \tag{2.3}$$

CNNs are biologically inspired as their local receptive fields mimic the visual cortex of the human brain, which leverage's local connectivity and spatial invariance

to excel at image analysis. Historically, small CNNs such as LeNet-5 by LeCun et al. (1998) succeeded in digit recognition, but the field's watershed moment came with AlexNet (Krizhevsky et al., 2012), which used a deep CNN to win the ImageNet challenge, igniting the deep learning era. CNNs soon became the de-facto standard in computer vision, including medical image analysis. In disease classification, a CNN trained on labeled medical images can automatically learn filters that detect pathological patterns (e.g., tumors, lesions) without needing hand-crafted features. This ability to learn multi-level features (edges $\rightarrow$ textures $\rightarrow$ anatomy $\rightarrow$ disease indicators) has led CNN-based models to achieve state-of-the-art performance in various diagnostics tasks.

2.7.2 VGG16 / VGG19

VGG models (Simonyan and Zisserman, 2014) use very small 3×3 convolutions stacked deep. VGG16 has 16 weight layers (13 conv + 3 FC), VGG19 has 19. Stacking multiple 3×3 layers increases receptive fields with fewer parameters than larger kernels. The convolution operation per layer is:

$$x_l = \sigma(W_l * x_{l-1} + b_l) \tag{2.4}$$

where σ is ReLU. Although VGG16 has ~138M parameters, making it heavy, it remains popular for transfer learning in disease classification.

2.7.3 ResNet (ResNet50 / ResNet101)

ResNets (He et al., 2015) introduced *residual connections* to allow very deep networks to train. A residual block computes:

$$y_l = x_l + F(x_l, W_l) \tag{2.5}$$

where $F(x_l, W_l)$ is the learned residual mapping. This solves the vanishing gradient problem, enabling networks with 50+ layers. ResNet50/101 use bottleneck blocks (1x1 $\rightarrow$ 3x3 $\rightarrow$ 1x1 conv) with identity skips or projection skips. ResNets are highly successful in disease detection because they capture features at multiple scales without degradation in accuracy.

2.7.4 MobileNetV2 / V3

MobileNet architectures are lightweight convolutional neural network designed for efficient computation on mobile and embedded devices without large accuracy loss. MobileNetV2 (2018) was introduced and its core architecture is the use of Depthwise separable convolutions and the inverted residual blocks with linear bottlenecks. Each block has:

1. a 1x1 expansion convolution that expands the channel dimension;

2. a Depthwise 3×3 convolution that performs spatial filtering on each channel;
3. a 1×1 projection convolution that reduces the channels back to a smaller number, yielding the block's output.

The output is:

$$y_m(i,j) = \sum_{c=1}^{C_{in}} P_{c,m} \left[(x_c * k_c)(i,j) \right] \tag{2.6}$$

where $x_c * k_c$ is the depthwise conv and $P_{c,m}$ is the pointwise mixing weight. This significantly reduces the number of parameters and computational cost while maintaining competitive accuracy.

MobileNetV3 (2019) further improved accuracy-latency tradeoffs using Neural Architecture Search (NAS), squeeze-and-excitation attention, and hard-swish activations. Both V2 and V3 are efficient, making them suitable for mobile plant disease detection.

2.7.5 InceptionV3

InceptionV3 (2016) is a deep CNN architecture designed to capture multi-scale features by applying multiple convolution filters of different sizes in parallel within the same layer. By combining these parallel feature maps, Inception achieves strong representational power, making it effective for complex image classification tasks such as plant disease detection, although at a higher computational cost compared to lightweight models. If x is the input, then a typical Inception module computes:

$$\text{Inception}(x) = [f_{1\times1}(x),\, f_{3\times3}(x),\, f_{5\times5}(x),\, f_{3\times3}^{pool}(x)] \tag{2.7}$$

where outputs are concatenated along channels. Factorized convolutions (e.g., 3×3 into $3\times1 + 1\times3$) reduce cost. InceptionV3 captures multi-scale features, useful in detecting diseases with varying lesion sizes.

2.7.6 Xception

Xception (Chollet, 2017) stands for "Extreme Inception" and is an Inception-inspired architecture that pushes the concept of filter factorization to its logical extreme. Xception replaces Inception module entirely by Depthwise separable convolutions arranged in a sequential stack of modules combined with linear residual connections.

It has 36 convolutional layers structured into an entry flow, middle flow, and exit flow. This design allows the model to learn highly expressive features while improving training efficiency. Mathematically, a Depthwise separable convolution in Xception is described by the formula:

$$y_m(i,j) = \sum_{c} P_{c,m} \left[(x_c * k_c)(i,j) \right] \tag{2.8}$$

This decouples spatial and cross-channel correlations entirely.

Xception has comparable parameters to InceptionV3 but achieves better performance, especially on large datasets. It is effective for plant disease classification due to efficient feature learning.

2.7.7 EfficientNet (B0–B7)

EfficientNet (Tan & Le, 2019) scales depth, width, and resolution uniformly with a compound factor ϕ:

$$d = \alpha^{\phi}, \quad w = \beta^{\phi}, \quad r = \gamma^{\phi} \tag{2.9}$$

with $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$. EfficientNet-B0 (5M params) up to B7 (66M params) achieve state-of-the-art accuracy-efficiency balance. They use MBConv blocks with squeeze-and-excitation. EfficientNet models excel in both lightweight and high-accuracy disease classification scenarios.

2.7.8 DenseNet (DenseNet121 / 169)

DenseNet (Huang et al., 2017) introduced dense connectivity: each layer receives inputs from all previous layers:

$$x_l = H_l([x_0, x_1, \ldots, x_{l-1}]) \tag{2.10}$$

where $[\cdot]$ denotes concatenation. This improves gradient flow, encourages feature reuse, and reduces parameters. DenseNet121/169 perform strongly on medical and plant disease datasets due to rich feature propagation.

Overall, each architecture balances depth, efficiency, and feature reuse differently. For plant disease classification, lightweight models like MobileNet and EfficientNet enable mobile deployment, while deeper models like ResNet, Xception, and DenseNet offer maximal accuracy.

2.8 Hybrid Model Classifiers

While traditional deep learning architectures such as VGG, ResNet, Inception, and MobileNet have shown impressive results in plant disease classification, hybrid model classifiers have emerged as a promising approach to balance accuracy, robustness, and efficiency. A hybrid model combines complementary features of multiple neural network architectures or integrates additional modules, such as attention mechanisms, to enhance feature extraction and classification performance. The motivation for hybridization stems from the limitations of single architectures—for example, lightweight models like MobileNet excel in efficiency but may underperform in complex classification tasks, while deeper networks like ResNet or Xception achieve higher accuracy but at the expense of computational cost.

One common hybrid strategy is the integration of attention mechanisms into baseline CNN architectures. Modules such as the Convolutional Block Attention Module (CBAM) and Squeeze-and-Excitation (SE) recalibrate feature maps by emphasizing relevant regions (e.g., diseased leaf lesions) and suppressing background noise [19, 44]. For instance, MobileNetV2 augmented with CBAM can maintain a lightweight structure while significantly improving focus on critical disease symptoms, leading to higher accuracy in plant disease detection without a substantial increase in model complexity.

Another hybridization approach is feature-level fusion, where outputs from multiple pre-trained models are combined. For example, features extracted from ResNet and Inception modules can be concatenated before the classification layer, leveraging ResNet's residual learning and Inception's multi-scale feature extraction. Such fusion-based hybrids often outperform single models in benchmark tasks due to richer feature representations [46]. Ensemble-based hybrids, which average or weight predictions from multiple networks, also enhance classification stability and reduce the variance of predictions.

In the context of plant disease classification, hybrid models have demonstrated superior performance over conventional single-architecture approaches. Recent studies show that hybrids integrating lightweight CNNs with attention modules or combining multiple deep networks consistently achieve accuracy improvements of 2–5% compared to their individual counterparts [30]. Importantly, hybrid classifiers can be optimized for mobile deployment by carefully choosing efficient backbones (e.g., MobileNet, EfficientNet) and lightweight attention blocks, thus providing a practical balance between diagnostic accuracy and computational feasibility.

2.9 Multi-Disease Classification and Transfer Learning

Plant leaves are susceptible to a wide variety of bacterial, viral, fungal, and abiotic disorders that present diverse symptoms across multiple crop species. Designing an automated system capable of recognizing *multiple diseases across multiple plant types* presents a non-trivial challenge due to inter-class similarities, intra-class variability, and data imbalance issues. Traditional classifiers often struggle with such variability, especially when trained on small or single-crop datasets.

Recent advancements in deep learning have enabled more robust approaches to multi-disease classification by leveraging large image datasets and sophisticated feature extractors. Models like CNNs are capable of learning hierarchical representations of leaf textures, color gradients, and lesion patterns that generalize well across disease types. However, training deep models from scratch typically requires massive amounts of labeled data—something not always available in agriculture.

This gap is bridged by *transfer learning*, a technique where models pretrained on large-scale datasets such as ImageNet are fine-tuned on specific tasks like plant disease identification. Transfer learning significantly reduces training time, mitigates

overfitting, and improves convergence on smaller agricultural datasets [16, 21]. For instance, Sladojevic *et al.* applied a pretrained CNN to classify 13 crop diseases with high accuracy, even with relatively limited training data [15].

This thesis leverages transfer learning by fine-tuning MobileNetV2, Xception, and Inception models pretrained on ImageNet to classify 38 disease classes from the PlantVillage dataset. The results demonstrate that with adequate augmentation and architectural tuning, lightweight models can achieve high performance in complex multi-class classification settings.

2.10 Depthwise Separable Convolutions

Depthwise separable convolutions (DSCs) are a fundamental innovation in modern deep learning architectures aimed at reducing computational cost and model complexity without significantly sacrificing accuracy. Unlike standard convolutional layers that apply a filter across all input channels simultaneously, DSCs decouple the filtering and combining steps into two separate operations: *depthwise convolution* and *pointwise convolution* [4].

In a standard 2D convolution layer, the computational cost is given by:

$$\text{Cost}_{\text{standard}} = D_K \times D_K \times M \times N \times D_F \times D_F \tag{2.11}$$

where:

- $D_K \times D_K$ is the kernel size,
- M is the number of input channels,
- N is the number of output channels,
- $D_F \times D_F$ is the spatial dimension of the feature map.

In contrast, a depthwise separable convolution factorizes the operation into:

- *Depthwise convolution:* Applies a single $D_K \times D_K$ filter to each input channel separately.
- *Pointwise convolution:* Applies a 1×1 convolution to combine the outputs from the depthwise step.

The resulting computational cost becomes:

$$\text{Cost}_{\text{DSC}} = D_K \times D_K \times M \times D_F \times D_F + M \times N \times D_F \times D_F \tag{2.12}$$

The ratio of computational reduction is:

$$\frac{\text{Cost}_{\text{DSC}}}{\text{Cost}_{\text{standard}}} = \frac{1}{N} + \frac{1}{D_K^2} \tag{2.13}$$

For typical values such as $D_K = 3$ and $N \gg 1$, the reduction is close to a factor of 8–9×, making it ideal for mobile and embedded applications [3].

Both MobileNetV2 and Xception architectures adopt this principle to create efficient yet powerful feature extractors. MobileNetV2 employs DSCs throughout the network, enabling fast inference on resource-constrained devices while preserving accuracy. Xception extends the idea further by replacing all standard convolutions with depthwise separable ones in an extreme form, achieving better performance with fewer parameters.

In this thesis, depthwise separable convolutions form the core of the proposed architecture, enabling efficient disease classification directly on mobile or edge devices. The reduced computational burden allows deployment in real-world agricultural environments without compromising speed or precision.

2.11 Attention Mechanism

In recent years, attention mechanisms have emerged as a cornerstone in deep learning models, especially in tasks requiring selective focus over sequential or spatial data. Originating in the field of natural language processing, attention enables models to dynamically weigh the importance of input features, thereby improving their capacity to model long-range dependencies and structural patterns. In plant disease detection, attention can enhance model interpretability and performance by allowing the model to focus on key symptomatic regions of diseased leaves.

2.11.1 Transformations of the Query, Key, and Value

The core of the attention mechanism lies in three learnable vectors: the Query **(Q)**, Key **(K)**, and Value **(V)**. These vectors are derived from the input feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ using linear transformations:

$$\mathbf{Q} = XW^Q, \quad \mathbf{K} = XW^K, \quad \mathbf{V} = XW^V \tag{2.14}$$

where $W^Q, W^K, W^V \in \mathbb{R}^{d \times d_k}$ are learnable weight matrices. Each row of $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ corresponds to the query, key, and value vector of a particular position in the input sequence or image region.

2.11.2 Calculation of Attention Scores

The attention score between two positions is calculated using the dot-product of their corresponding query and key vectors:

$$\text{Score}(i, j) = \mathbf{Q}_i \cdot \mathbf{K}_j^\top \tag{2.15}$$

This results in an $n \times n$ score matrix representing the similarity between each query and all keys.

2.11.3 Scaled Dot-Product Attention

To stabilize the gradient and avoid extremely large dot products, the raw scores are scaled by the square root of the key dimension d_k, followed by a softmax to obtain normalized weights:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.16}$$

where:

- Q = Query matrix,
- K = Key matrix,
- V = Value matrix,
- d_k = Dimensionality of the keys.

This operation assigns higher weights to relevant positions, allowing the model to attend selectively across the input.

2.11.4 Self-Attention Mechanism

In self-attention, the same input sequence is used to compute the queries, keys, and values. Thus, each token attends to every other token, including itself. This allows contextual information to flow across all positions:

$$\text{SelfAttention}(X) = \text{softmax}\left(\frac{XW^Q(XW^K)^\top}{\sqrt{d_k}}\right)XW^V \tag{2.17}$$

2.11.5 Multi-Head Attention

Instead of performing a single attention operation, Multi-Head Attention (MHA) runs multiple self-attention blocks in parallel and concatenates the outputs:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2.18}$$

where each attention head is computed as:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.19}$$

Here, W_i^Q, W_i^K, W_i^V are projections for the i-th head, and W^O is the output projection. MHA enables the model to jointly attend to information from different representation subspaces, enhancing robustness and expressiveness.

2.11.6 Application in Plant Disease Detection

In plant pathology, attention mechanisms—especially self-attention and CBAM (Convolutional Block Attention Module)—have been effective in highlighting key disease regions, even under complex backgrounds or varying lighting conditions. By integrating such mechanisms, lightweight CNNs can be enhanced to improve both accuracy and interpretability, aligning well with mobile deployment goals. Studies show that attention-enhanced CNNs outperform baseline models in classification accuracy, precision, and interpretability [44, 45]. In this thesis, the CBAM module is integrated into the MobileNetV2 backbone, and experimental results confirm that the attention-enhanced architecture yields higher classification accuracy on the PlantVillage dataset compared to its non-attentive counterpart.

2.12 Convolutional Block Attention Module (CBAM)

The Convolutional Block Attention Module (CBAM) is a lightweight yet powerful attention mechanism designed to improve the representational power of convolutional neural networks by emphasizing informative features while suppressing less useful ones [44]. CBAM achieves this by sequentially applying two types of attention: *channel attention* and *spatial attention* as shown in Figure 2.1.

2.12.1 Architecture Overview

CBAM takes an intermediate feature map $F \in \mathbb{R}^{C\times H\times W}$ and produces a refined output $\hat{F}$ via:

$$\hat{F} = M_s(M_c(F) \cdot F) \cdot (M_c(F) \cdot F) \tag{2.20}$$

where $M_c(F)$ and $M_s(\cdot)$ represent the channel and spatial attention maps, respectively, and $\cdot$ denotes element-wise multiplication.

Channel Attention Module (CAM)

The channel attention module determines *what* features are important by computing:

$$M_c(F) = \sigma(\text{MLP}(\text{AvgPool}(F)) + \text{MLP}(\text{MaxPool}(F))) \tag{2.21}$$

where:

- AvgPool(F) and MaxPool(F) are global pooling operations applied across spatial dimensions,
- MLP is a shared multi-layer perceptron with one hidden layer,
- σ is the sigmoid activation function.

This module captures global contextual dependencies across channels.

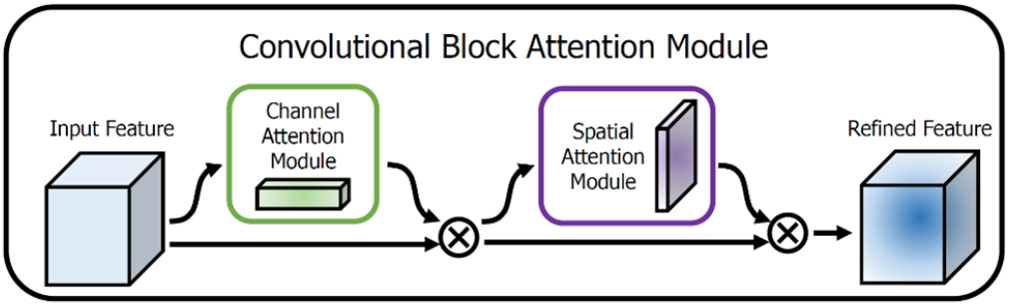

Figure 2.1: The CBAM architecture [44].

Spatial Attention Module (SAM)

The spatial attention module determines *where* to focus by computing a 2D attention map:

$$M_s(F') = \sigma(f^{7\times7}([\text{AvgPool}(F'); \text{MaxPool}(F')])) \tag{2.22}$$

where:

- F' is the feature map refined by the channel attention,
- $f^{7\times7}$ is a convolutional layer with a 7×7 kernel,
- $[\cdot;\cdot]$ denotes concatenation along the channel axis.

This step captures spatially important regions within the image.

2.12.2 Application to Plant Disease Detection

In plant disease classification, symptoms often appear as localized discoloration, spots, or texture changes that may be subtle and vary by class. CBAM improves the ability of the network to:

- Focus on disease-relevant regions (e.g., leaf lesions, blight edges),
- Suppress background or irrelevant features (e.g., stems, soil),
- Improve localization even under variations in lighting, occlusion, or shape.

In this thesis, CBAM is integrated into the MobileNetV2 architecture, forming the hybrid MobileNetV2 + CBAM model. Experimental results demonstrate that CBAM-enhanced models consistently outperform their plain CNN counterparts, achieving better accuracy, precision, and Grad-CAM focus on disease regions.

2.13 IoT and Edge Computing Integration

The integration of the Internet of Things (IoT) and edge computing has significantly enhanced the applicability of artificial intelligence in agriculture, particularly for plant disease monitoring. IoT-based agricultural systems consist of interconnected sensors, devices, and microcontrollers that collect environmental data—such as humidity, temperature, and soil conditions—to support precision farming decisions. However, relying solely on cloud-based processing introduces latency, energy consumption, and dependency on network availability, which are problematic in remote agricultural areas.

Edge computing addresses these limitations by enabling computation to occur locally, directly on the device or near the data source. In the context of plant disease detection, edge devices such as *smartphones*, *Jetson Nano*, *Raspberry Pi*, and other embedded systems can now run lightweight deep learning models trained for classification tasks. This capability supports real-time inference, even in offline environments, and empowers farmers with immediate feedback. For instance, Ramcharan *et al.* developed a CNN-powered Android application capable of identifying cassava leaf diseases in the field without needing cloud support [33]. Similarly, Howard *et al.* introduced *MobileNet*, an architecture optimized for mobile and edge devices through the use of *depthwise separable convolutions*, reducing model size while retaining competitive accuracy [4].

Recent studies have demonstrated that *edge-deployed models*, especially those trained with *transfer learning* and attention mechanisms like CBAM, can effectively detect diseases with high accuracy and low latency [44, 46]. Integrating such models into mobile phones or microcontrollers using platforms like TensorFlow Lite, Core ML, or NVIDIA JetPack SDK makes them practical for on-site decision-making. This thesis aligns with these trends by proposing a hybrid lightweight deep learning architecture (MobileNetV2+CBAM) trained on the PlantVillage dataset and evaluated for deployment on mobile devices. By combining model efficiency, high classification accuracy, and edge compatibility, the proposed system supports a future of smart agriculture where disease detection is accessible, scalable, and real-time.

2.14 Spectral & Hyperspectral Imaging for Disease Detection

Spectral and hyperspectral imaging (HSI) techniques have gained attention in precision agriculture for their ability to detect subtle physiological and biochemical changes in plant tissues before visual symptoms appear. Unlike conventional RGB imaging, which captures data in three broad bands (red, green, blue), hyperspectral sensors collect data across *hundreds of narrow spectral bands*, enabling the identification of disease-specific spectral signatures at early stages [11].

In spectral imaging, each pixel contains detailed spectral reflectance information, allowing the distinction between healthy and diseased tissue based on differences in light absorption and scattering properties. This capability makes HSI particularly useful in detecting fungal infections, viral symptoms, and nutrient deficiencies that might not be evident to the human eye or RGB cameras [37]. Mahlein *et al.* showed that imaging sensors combined with machine learning algorithms could identify sugar beet diseases using visible and near-infrared spectra with high accuracy [11]. Similarly, studies using HSI to monitor wheat rust and citrus greening disease achieved early-stage detection and high classification precision in controlled experiments [24].

However, despite their diagnostic potential, spectral and hyperspectral systems face practical constraints for real-world deployment. These include *high cost*, *large data volume*, *sensor complexity*, and *computational overhead* for real-time inference. Furthermore, spectral cameras are typically *not available on mobile devices*, limiting their use in low-resource settings.

In contrast, this thesis adopts a deep learning–based approach using RGB imagery, which is more accessible for farmers through smartphone cameras and portable edge devices. By leveraging RGB images from the publicly available PlantVillage dataset and optimizing the model architecture for mobile deployment, the proposed system balances performance with real-world feasibility—while remaining extensible to future integration with multispectral inputs.

2.15 Responsible AI (RAI) and Explainable AI (XAI) in Plant Disease Detection

As artificial intelligence continues to shape high-impact domains such as agriculture, the need for ethical, transparent, and accountable AI has never been greater. Responsible AI (RAI) encompasses the principles of fairness, safety, privacy, transparency, and accountability in the design and deployment of AI systems. In the context of plant disease detection, these principles are critical to ensure that farmers, agronomists, and policymakers can trust the system outputs and interpret decisions that affect crop management and food security.

Explainable AI (XAI), a core aspect of RAI, focuses on making model predictions intelligible to human users. Most deep learning architectures, despite their high predictive accuracy, are considered "black-box" models. This opacity poses a significant limitation in agriculture, where stakeholders often require explanations for why a model identified a plant as diseased, especially in high-stakes decisions involving pesticide use, quarantine, or yield estimation.

Various XAI methods, such as saliency maps, SHAP (SHapley Additive exPlanations), and Grad-CAM (Gradient-weighted Class Activation Mapping), have been adopted in agricultural AI systems. For instance, Grad-CAM generates visual

heatmaps that highlight regions of plant leaves contributing to a specific classification [38]. This not only builds user confidence but also helps validate the model's focus on biologically relevant features, such as lesions or discoloration patterns. In this thesis, Grad-CAM is applied to evaluate whether models like MobileNetV2 and Xception focus on disease-affected regions, particularly before and after attention modules such as CBAM are introduced.

Chapter 3

Methodology

3.1 Overview of Methodology

This study proposes an efficient and scalable deep learning pipeline for the detection and classification of plant leaf diseases using RGB images. The methodology follows a modular pipeline consisting of the following major stages: (1) dataset collection and understanding, (2) preprocessing and augmentation, (3) feature extraction via deep learning models, (4) integration of attention mechanisms to improve discriminability, (5) model evaluation and visualization, and (6) deployment to mobile/edge environments for real-time classification.

The pipeline leverages the publicly available PlantVillage dataset comprising 54,306 images across 38 distinct plant disease classes. These images are split into training, validation, and testing sets using an 80:10:10 ratio. Image augmentation techniques, including rotation, zoom, and horizontal flips, are applied to mitigate overfitting and improve generalization.

Several deep convolutional neural network (CNN) architectures are explored, including MobileNetV2, InceptionV3, and Xception. A hybrid variant that combines MobileNetV2 with the Convolutional Block Attention Module (CBAM) is proposed as the core of the final model. All models are trained using transfer learning with ImageNet weights and optimized using the Adam optimizer with a learning rate scheduler and early stopping.

Model performance is evaluated using classification metrics such as accuracy, precision, recall, F1-score, AUC-ROC, and confusion matrices. Interpretability is enhanced through Grad-CAM visualizations that highlight disease-relevant regions on leaves. The entire pipeline is implemented using TensorFlow, Keras, and scikit-learn libraries in Python. Final deployment considerations include quantization and export to TensorFlow Lite for on-device inference in resource-constrained environments such as smartphones or edge devices.

Table 3.1: Sample distribution of selected classes from the PlantVillage dataset

Crop	Disease	Image Count
Apple	Apple Scab	630
Grape	Black Rot	1,180
Potato	Late Blight	1,000
Tomato	Bacterial Spot	2,127
Squash	Powdery Mildew	1,835
Corn	Common Rust	1,192
Peach	Healthy	614

3.2 Dataset Description

The dataset used in this research is the *PlantVillage* dataset, a publicly available benchmark for plant disease classification tasks [20]. It contains a total of 54,306 RGB images of healthy and diseased plant leaves, distributed across 38 distinct classes covering 14 crop species and 26 different diseases. Each image in the dataset is annotated with the crop name and the corresponding disease label (or "healthy" if applicable).

The images were originally collected under controlled lighting conditions, with plain backgrounds to isolate leaf features. The resolution of images varies, but all were resized to 224×224 pixels during preprocessing to match the input shape of the selected CNN architectures.

3.2.1 Class Distribution and Labeling

Table 3.1 shows a representative breakdown of several crop–disease classes and the number of samples in each. Although some classes such as Tomato and Apple have multiple disease subtypes, others like Corn have fewer examples, creating a mild class imbalance. This imbalance is addressed using image augmentation during training.

3.2.2 Data Splitting Strategy

The dataset was randomly divided into training (80%), validation (10%), and test (10%) subsets, ensuring representative class distribution across all splits. Stratified sampling was applied to preserve class balance. No image appeared in more than one subset, preventing direct data leakage.

However, while duplicate images were avoided, it is possible that correlations exist between samples due to similar acquisition conditions, such as images captured from the same plant or under similar environmental settings (e.g., lighting, background, or camera conditions). Such correlations may introduce a degree of

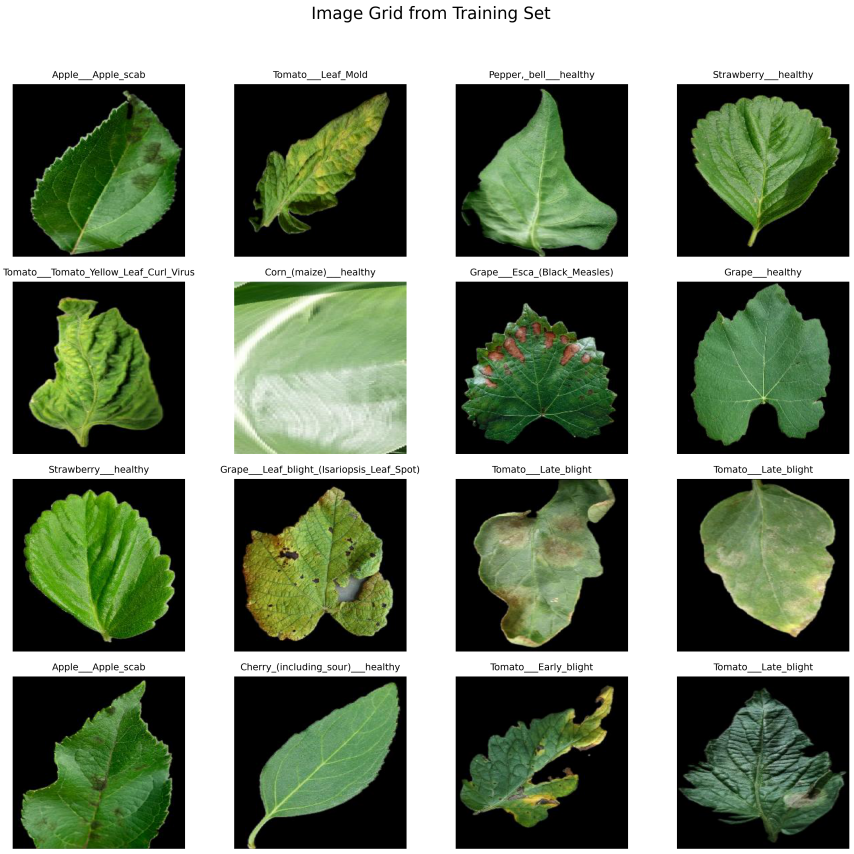

Figure 3.1: Sample images from different classes in the PlantVillage dataset.

bias and could lead to a slight overestimation of the model's generalization performance. This limitation is inherent to the dataset and should be considered when interpreting the results. All preprocessing and data loading were implemented using `{tensorflow.keras.preprocessing}` and `{ImageDataGenerator}`, ensuring reproducibility and compatibility with the model training pipeline.

3.2.3 Exploratory Visualization

To provide a qualitative overview of the dataset, Figure 3.1 presents a 4×4 grid of sample images randomly selected from various classes. Each image has been resized and padded for uniform display. This provided a visual understanding of dataset diversity, disease symptoms, and image quality.

3.3 Data Preprocessing

Preprocessing plays a critical role in preparing raw image data for effective deep learning. To ensure uniformity, generalizability, and robust feature learning, several preprocessing steps were applied to the PlantVillage dataset prior to training.

3.3.1 Image Resizing and Normalization

All images were resized to 224×224 pixels, matching the input dimensions expected by MobileNetV2, Xception, and InceptionV3 models. Resizing ensures consistent spatial dimensions across the dataset while maintaining key visual patterns such as disease spots or blights. The pixel values were then normalized to a $[0, 1]$ range by dividing each channel by 255.

$$x_{\text{norm}} = \frac{x_{\text{pixel}}}{255} \tag{3.1}$$

This normalization speeds up convergence and stabilizes the gradients during back-propagation.

3.3.2 Image Augmentation

To address class imbalance and improve model generalization, real-time image augmentation was applied using the `{ImageDataGenerator}` module from `[tensorflow.keras.preprocessing.image]` library in the TensorFlow framework. Augmentation techniques simulate variations such as rotation, scaling, and flipping that may occur in real-world plant leaf imagery.

- `rotation_range = 25` — randomly rotates the image by up to 25 degrees.
- `zoom_range = 0.2` — randomly zooms inside the image.
- `width_shift_range = 0.2` — performs minor translations.
- `height_shift_range = 0.2` — performs minor translations.
- `horizontal_flip = True` — flips the image horizontally.
- `shear_range = 0.15` — applies affine transformations.

These transformations increase the diversity of the training set, which helps prevent overfitting and improves the model's ability to generalize to unseen data.

3.3.3 Label Encoding

The disease class names were one-hot encoded using `LabelBinarizer` from `scikit-learn` to produce a C-dimensional binary vector for each class, where $C = 38$ is the total number of unique classes in the dataset.

$$y_i = [0, \ldots, 0, 1, 0, \ldots, 0], \quad i = \text{class index} \tag{3.2}$$

This encoding is required for training with categorical cross-entropy loss.

3.3.4 Train/Validation/Test Pipeline

Three separate `ImageDataGenerator` instances were used for the training, validation, and test sets. Augmentation was applied only to the training generator, while the validation and test sets used only resizing and normalization to ensure consistent performance evaluation.

All images were loaded in batches using the `flow_from_directory()` function, with class labels automatically inferred from folder names.

3.4 Model Architecture

This research investigates multiple deep convolutional neural network (CNN) architectures for efficient plant disease classification. The models selected for experimentation include MobileNetV2, InceptionV3, Xception, and a proposed hybrid model: MobileNetV2 integrated with a Convolutional Block Attention Module (CBAM). All architectures utilize transfer learning from pretrained ImageNet weights and are fine-tuned on the PlantVillage dataset.

3.4.1 MobileNetV2

MobileNetV2 is a lightweight deep network optimized for mobile and embedded vision applications. It builds upon the efficiency of MobileNetV1 by introducing *inverted residual blocks* and *linear bottlenecks* [4]. The core component is the depthwise separable convolution, which reduces computational cost by factorizing a standard convolution into a depthwise convolution followed by a 1×1 pointwise convolution.

The MobileNetV2 base was loaded with pretrained weights and fine-tuned by unfreezing the last 30 layers. A global average pooling layer, followed by dense layers with dropout regularization and softmax output, was appended to adapt it for the 38-class classification task.

3.4.2 InceptionV3

InceptionV3 employs a factorized convolution strategy that reduces the number of parameters while capturing multi-scale features. The architecture introduces

inception modules that apply 1×1, 3×3, and 5×5 convolutions in parallel, capturing both local and global patterns [6]. In this study, the InceptionV3 base was loaded with ImageNet weights and appended with a custom classification head.

3.4.3 Xception

Xception stands for "Extreme Inception" and is a depthwise separable variant of the Inception family [3]. Instead of concatenating outputs from different kernel sizes, Xception replaces all convolutions with depthwise separable convolutions. This architecture achieves strong performance with fewer parameters and was evaluated in the same pipeline as the other models.

3.4.4 Proposed Hybrid Model: MobileNetV2 + CBAM

To enhance spatial and channel feature representation, this work proposes a hybrid model that integrates the Convolutional Block Attention Module (CBAM) into MobileNetV2. CBAM refines the intermediate feature maps by sequentially applying:

- *Channel Attention* via global max and average pooling and a shared MLP.
- *Spatial Attention* via 7×7 convolution on concatenated spatial descriptors.

The CBAM module was inserted after the final bottleneck block of MobileNetV2. This design enhances the network's ability to focus on disease-relevant regions, such as localized leaf lesions, while suppressing background noise.

3.4.5 Unified Classification Head

All models shared a consistent classification head for comparability. The head consists of:

- Global Average Pooling (GAP)
- Dense layer with ReLU activation
- Dropout (0.4)
- L2 regularization ($\lambda = 0.0005$)
- Final Dense layer with 38 output nodes and softmax activation

3.4.6 Parameter and Complexity Summary

Table 3.2 summarizes the number of trainable parameters and the approximate model size after fine-tuning.

Table 3.2: Comparison of model parameters and size

Model	Trainable Parameters	Model Size (MB)
MobileNetV2	3.6M	36.2
MobileNetV2 + CBAM	4.0M	40.9
Xception	22.9M	172.7
InceptionV3	23.9M	147.8

3.5 Training Configuration

All deep learning models in this study were implemented and trained using the TensorFlow and Keras libraries. The training process was standardized across all architectures to ensure fair and consistent performance comparison. This section outlines the optimizer settings, learning rate schedule, batch size, epochs, and regularization strategies used throughout the training pipeline.

3.5.1 Optimizer and Learning Rate

The Adam optimizer was employed for training all models, due to its adaptive learning rate properties and robustness to sparse gradients [22]. The initial learning rate was set to 1×10^{-5} and kept constant throughout training. In addition, a `ReduceLROnPlateau` callback was configured to reduce the learning rate by a factor of 0.2 if the validation accuracy plateaued for 5 consecutive epochs.

$$\text{Adam Update:} \quad \theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t \tag{3.3}$$

where $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected first and second moment estimates, respectively.

3.5.2 Batch Size and Epochs

A batch size of 32 was used across all models to balance training efficiency with GPU memory limitations. Each model was trained for a maximum of 30 epochs, with early stopping applied to terminate training if the validation accuracy did not improve for 6 consecutive epochs. This approach prevents overfitting and unnecessary computation.

3.5.3 Regularization Techniques

Two regularization techniques were applied to mitigate overfitting:

- **Dropout:** A dropout rate of 0.4 was applied after the penultimate dense layer in the classification head to randomly deactivate neurons during training.

- **L2 Regularization:** An L2 penalty term $\lambda = 0.0005$ was optionally added to the kernel weights of dense layers, particularly for deeper models like InceptionV3 and Xception.

$$L_{\text{total}} = L_{\text{cross-entropy}} + \lambda \sum_{i=1}^{n} w_i^2 \tag{3.4}$$

3.5.4 Callbacks and Checkpoints

Three training callbacks were used to manage training efficiency:

- `ModelCheckpoint`: Saved the best model weights based on validation accuracy.
- `EarlyStopping`: Monitored validation loss and stopped training after 6 stagnant epochs.
- `ReduceLROnPlateau`: Dynamically reduced the learning rate upon validation stagnation.

All experiments were executed on a dedicated GPU setup, ensuring fast training cycles and reproducibility.

3.6 Evaluation Metrics

To comprehensively assess the performance of the proposed models, this study employs multiple classification metrics beyond simple accuracy. These include precision, recall, F1-score, confusion matrix, and the area under the receiver operating characteristic curve (AUC-ROC). The use of multiple evaluation criteria is crucial in multi-class disease classification, where class imbalance and misclassification can lead to misleading results if accuracy alone is considered.

3.6.1 Accuracy

Accuracy measures the proportion of correct predictions over the total number of predictions made:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.5}$$

For multi-class classification, categorical accuracy is used, which compares the predicted class label against the true label across all classes.

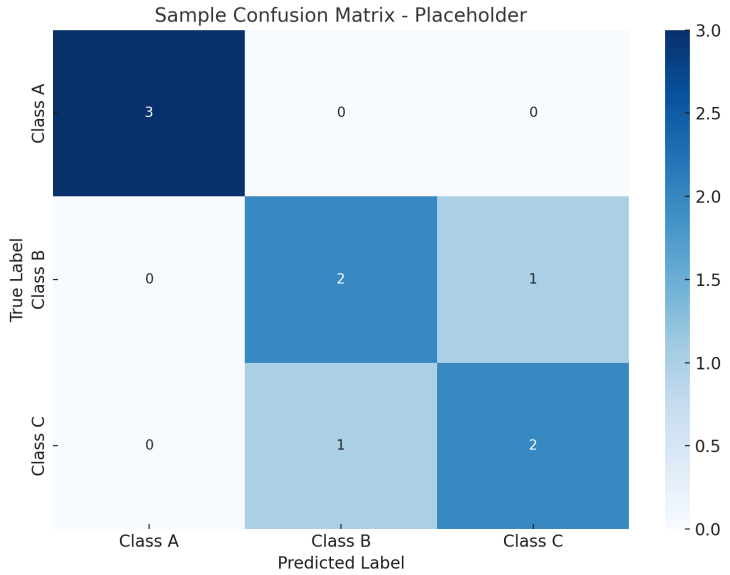

Figure 3.2: Sample confusion matrix for the proposed model.

3.6.2 Precision, Recall, and F1-Score

Precision and recall are particularly relevant in agricultural applications where false positives (e.g., misdiagnosing a healthy leaf) and false negatives (e.g., missing a disease) can have different consequences.

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN} \tag{3.6}$$

The F1-score provides a harmonic mean between precision and recall:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.7}$$

These metrics are calculated per class and averaged using the macro or weighted averaging strategy.

3.6.3 Confusion Matrix

A confusion matrix is used to visualize the classification performance by showing the number of true positive, false positive, true negative, and false negative predictions for each class. This matrix helps identify which classes are frequently misclassified as shown in Figure 3.2.

3.6.4 AUC-ROC Score

The AUC-ROC metric evaluates a model's ability to distinguish between classes across all thresholds. In multi-class classification, a one-vs-rest approach is adopted, and the AUC is averaged across all classes:

$$\text{AUC} = \int_0^1 \text{TPR}(x)\, dx \tag{3.8}$$

where TPR is the true positive rate and the curve plots TPR against the false positive rate (FPR). A higher AUC indicates better separability between classes.

3.6.5 Visualization and Reporting

All metrics were computed using `scikit-learn`'s `classification_report`, and `roc_auc_score` functions. Performance plots, including per-class bar charts and ROC curves, were generated for visual insight into class-wise model behavior.

3.7 Visualization Tools

To enhance interpretability and facilitate the analysis of model behavior, various visualization techniques were integrated into the experimental pipeline. These tools not only provide quantitative insights but also offer qualitative validation by visualizing the decision-making process of the models.

3.7.1 Grad-CAM (Gradient-weighted Class Activation Mapping)

Grad-CAM is an explainable AI technique used to produce visual explanations for CNN-based models by highlighting image regions that are most influential in predicting a specific class [38]. This is particularly useful for plant disease detection, where it is important to ensure that the model focuses on disease-affected regions of the leaf rather than background noise.

The Grad-CAM heatmap $L^{\text{Grad-CAM}}$ is computed as:

$$L^{\text{Grad-CAM}} = \text{ReLU}\left(\sum_k \alpha_k A^k\right) \tag{3.9}$$

where:

- A^k is the k-th feature map of the last convolutional layer,
- α_k is the weight computed by global average pooling the gradient of the score for class c with respect to feature map A^k,

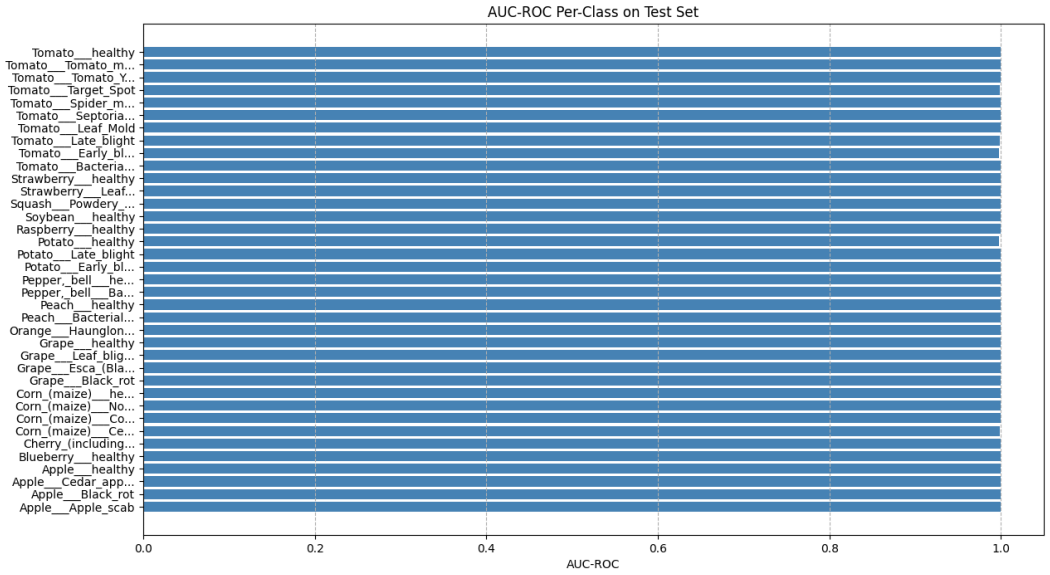

Figure 3.3: Example ROC curves for selected classes.

- ReLU ensures that only positively contributing features are visualized.

Grad-CAM was used to visualize predictions for both MobileNetV2 and MobileNetV2+CBAM models. Comparison of attention maps before and after CBAM integration demonstrated improved focus on lesion areas and symptom zones, thus illustrating the effectiveness of the attention mechanism.

3.7.2 ROC Curves and AUC Plots

Receiver Operating Characteristic (ROC) curves were generated for each class using a one-vs-rest strategy. The curves were plotted using scikit-learn's `roc_curve` and `auc` functions. An example ROC plot is shown in Figure 3.3, comparing AUC performance across multiple classes.

3.7.3 Per-Class Accuracy and AUC Bar Charts

To summarize class-wise performance, bar charts were created showing either per-class accuracy or AUC scores. These visualizations aid in identifying classes with high or low confidence and highlight performance discrepancies across different disease types as shown in Figure 3.4.

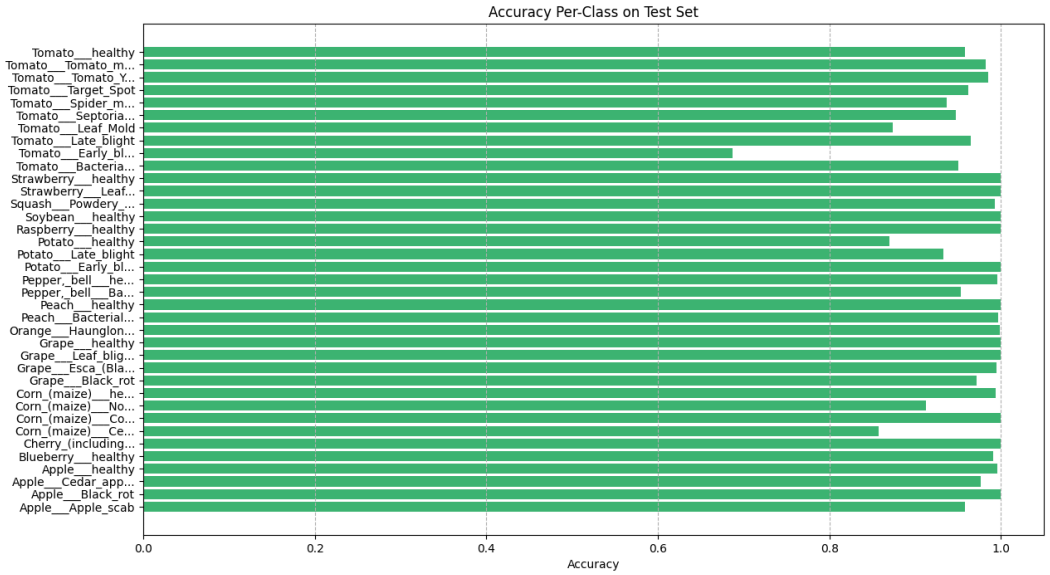

Figure 3.4: Per-class classification accuracy for the proposed model.

3.8 Deployment Pipeline

The final objective of this research is to explore the feasibility of deploying the trained plant disease detection model onto mobile or edge devices for practical use. This requires converting the trained model into a lightweight format compatible with mobile platforms, such as Android smartphones or embedded systems. The deployment pipeline consists of model optimization, conversion to TensorFlow Lite format, and inference testing under resource-constrained conditions.

3.8.1 Model Optimization for Deployment

To reduce the model's memory footprint and improve inference speed, post-training quantization was applied using TensorFlow Lite's optimization tools. Specifically, dynamic range quantization was employed, which converts model weights from 32-bit floating point to 8-bit integers, thereby reducing model size and improving inference efficiency while maintaining acceptable accuracy.

- **Weight Quantization:** Converts 32-bit floating point weights to 8-bit integers, reducing model size and improving inference speed.
- **Integer-aware Inference:** Enables more efficient execution on CPU-based mobile inference engines.

It should be noted that other optimization techniques, such as model pruning and full integer quantization, were not implemented in this study but remain potential avenues for further reducing model size and improving deployment efficiency in future work.

3.8.2 TensorFlow Lite Conversion

The optimized model was converted to TensorFlow Lite format using the following steps:

1. Save the trained Keras model in `.h5` format.
2. Load the model and convert to `.tflite` using the TensorFlow Lite Converter:

```
model = tf.keras.models.load_model(model_path)
converter =
        tf.lite.TFLiteConverter.from_keras_model(model)
```

3. Transfer the converted model to the target device for testing.

3.8.3 Mobile Inference and Testing

The exported TensorFlow Lite model was deployed on an Android smartphone using TensorFlow Lite's Android support library. Inference was performed on test images using the device camera or file picker. A lightweight mobile interface was created with a simple user experience for selecting images and viewing predictions.

3.9 Summary

This chapter has presented a comprehensive overview of the methodology adopted for building an efficient and deployable deep learning system for plant disease classification. The proposed pipeline began with the acquisition and exploratory analysis of the PlantVillage dataset, followed by rigorous preprocessing steps including resizing, normalization, augmentation, and label encoding.

Multiple deep learning architectures namely, MobileNetV2, Xception, InceptionV3, as well as a hybrid model combining MobileNetV2 with CBAM were implemented and evaluated. Each model was fine-tuned using transfer learning and trained for 30 epochs with the Adam optimizer and a low learning rate to ensure convergence. Regularization techniques such as dropout and L2 penalties were incorporated to mitigate overfitting, while callbacks like early stopping and learning rate reduction helped optimize training efficiency.

Performance evaluation relied on robust metrics such as accuracy, precision, recall, F1-score, AUC-ROC, and confusion matrices. Advanced visualization tools

like Grad-CAM and per-class bar charts provided interpretability and transparency to model decisions. Finally, the most efficient model was optimized and converted to TensorFlow Lite format for mobile deployment, demonstrating real-world feasibility and high accuracy with minimal computational footprint.

This methodological framework sets the stage for detailed experimental analysis in the next chapter, where comparative results, error analysis, and insights into model behavior will be explored.

Chapter 4

Experimental Results and Analysis

4.1 Experimental Setup

This section outlines the hardware, software, and configuration details used during the model training and evaluation process. All experiments were conducted on a local workstation equipped with the following specifications:

- **GPU:** NVIDIA GeForce RTX 3050 Ti Laptop GPU (4 GB VRAM)
- **CPU:** Intel Core i7, 11th Gen (8 cores)
- **RAM:** 16 GB DDR4
- **Operating System:** Windows 11 Pro
- **Frameworks:** Python 3.10, TensorFlow 2.13, Keras 2.13, scikit-learn 1.3

Model development, training, and evaluation were performed using Jupyter notebooks in a TensorFlow-based environment. GPU acceleration was enabled via CUDA and cuDNN backends to speed up the training process, particularly for larger architectures such as Xception and InceptionV3.

All models were trained using the same hyperparameters for comparability:

- **Epochs:** 30
- **Batch Size:** 32
- **Optimizer:** Adam (learning rate = 1×10^{-5})
- **Loss Function:** Categorical Cross-Entropy
- **Callbacks:** EarlyStopping, ReduceLROnPlateau, ModelCheckpoint

Table 4.1: Model performance comparison on test set

Model	Accuracy	Precision	Recall	F1-Score	AUC
Inception *(CONE)*	0.9156 ± 0.003	0.9225 ± 0.004	0.9156 ± 0.003	0.9160 ± 0.003	0.9912 ± 0.001
MobileNetV2 *(RHOMBUS)*	0.9692 ± 0.002	0.9707 ± 0.002	0.9692 ± 0.002	0.9689 ± 0.002	0.9984 ± 0.001
MobileNetV2+CBAM *(CUBE)*	0.9732 ± 0.002	0.9747 ± 0.002	0.9732 ± 0.002	0.9731 ± 0.002	0.9987 ± 0.001
Xception *(PRISM)*	0.9830 ± 0.002	0.9833 ± 0.002	0.9830 ± 0.002	0.9830 ± 0.002	0.9995 ± 0.0005

Performance plots for each model were saved and analyzed using Matplotlib and Seaborn. Grad-CAM visualizations were generated using TensorFlow's gradient tape API to highlight attention focus regions during classification.

All four models—MobileNetV2, InceptionV3, Xception, and the proposed MobileNetV2 + CBAM—were trained using identical data splits and evaluation metrics for fairness.

4.2 Performance Metrics Summary

This section summarizes the key performance metrics for all four models trained and evaluated in this study: MobileNetV2 (RHOMBUS), MobileNetV2 + CBAM (CUBE), Xception (PRISM), and InceptionV3 (CONE). Each model was trained for 30 epochs using identical configurations. To provide a more reliable evaluation, model performance is reported as the mean and standard deviation across multiple independent training runs. The models were assessed using five core metrics: Accuracy, Precision, Recall, F1-score, and AUC Score.

4.2.1 Model Comparison Table

Table 4.1 presents the comparison of classification metrics for each model, reported as mean ± standard deviation over multiple runs. This provides an estimate of variability and allows a more reliable interpretation of performance differences between models.

The standard deviation provides an estimate of variability across runs. While the MobileNetV2+CBAM model shows improved performance compared to the baseline MobileNetV2, the observed differences are relatively small and fall within overlapping ranges of variability. Therefore, the improvement can be interpreted as a modest enhancement rather than a substantial performance gain.

Table 4.2: Experimental results reported by Mohanty et al. [28] for deep learning-based plant disease detection

Model	Image Type	Accuracy	Precision	Recall	F1-Score
AlexNet	Color	0.9928	0.9928	0.9927	0.9927
	Grayscale	0.9725	0.9728	0.9727	0.9726
	Segmented	0.9892	0.9893	0.9891	0.9891
GoogleNet	Color	0.9935	0.9935	0.9935	0.9934
	Grayscale	0.9798	0.9804	0.9801	0.9800
	Segmented	0.9924	0.9925	0.9925	0.9925

4.2.2 Model Comparison with Related Work

The experimental results reported by Mohanty et al. [28], summarized in Table 4.2, present a benchmark evaluation of deep convolutional neural networks for plant disease classification using the PlantVillage dataset. Their study evaluates architectures such as AlexNet and GoogLeNet across different image configurations, including color, grayscale, and segmented images, achieving high classification accuracy under controlled conditions. These results provide a strong reference point for comparing the performance of the models proposed in this study.

A comparison of classification accuracy between the models evaluated in this study (Table 4.1) and the results reported by Mohanty et al. [28] (Table 4.2) shows that the highest accuracy achieved in this work is 98.30% using the Xception model, while the MobileNetV2 + CBAM (CUBE) model achieved 97.32%. In contrast, Mohanty et al. reported higher accuracies of up to 99.35% using GoogLeNet on color images from the PlantVillage dataset. This difference in performance may be attributed to variations in experimental setup, model architectures, and training configurations, as well as the inherent characteristics of the dataset, which consists of images captured under controlled conditions.

Despite the slightly lower accuracy compared to the results of Mohanty et al., the models proposed in this study, particularly the MobileNetV2-based architectures, offer a significant advantage in terms of computational efficiency and suitability for resource-constrained environments. The integration of CBAM provides a modest improvement over the baseline MobileNetV2 model while maintaining its lightweight nature. Therefore, the contribution of this work lies not in surpassing the highest benchmark accuracy, but in achieving a strong balance between performance and efficiency, which is essential for practical deployment in real-world agricultural applications.

As shown in Table 4.3, while the models in this study achieve slightly lower accuracy than those reported by Mohanty et al., they provide a more favorable balance between performance and computational efficiency.

Table 4.3: Comparison of classification accuracy with Mohanty et al. [28]

Study / Model	Configuration	Accuracy
This Work		
Xception (PRISM)	Color Images	0.9830 ± 0.002
MobileNetV2 + CBAM (CUBE)	Color Images	0.9732 ± 0.002
MobileNetV2 (RHOMBUS)	Color Images	0.9692 ± 0.002
InceptionV3 (CONE)	Color Images	0.9156 ± 0.003
Mohanty et al. (2016)		
GoogLeNet	Color Images	0.9935
AlexNet	Color Images	0.9928
GoogLeNet	Segmented Images	0.9924
AlexNet	Segmented Images	0.9892

Table 4.4: Training duration of models (30 Epochs)

Model	Codename	Training Time (HH:MM:SS)
MobileNetV2	RHOMBUS	01:56:05
MobileNetV2 + CBAM	CUBE	01:56:49
Xception	PRISM	02:09:26
InceptionV3	CONE	04:09:26

4.2.3 Training Duration

The total training time for each model over 30 epochs was recorded on a system equipped with an NVIDIA GeForce RTX 3050 Ti GPU and 16GB RAM. Table 4.4 summarizes the training duration for each model.

As shown in Table 4.4, the MobileNetV2 and MobileNetV2+CBAM models exhibit comparable training times, indicating that the integration of the CBAM module introduces only a minimal computational overhead. In contrast, the Xception and InceptionV3 models require longer training durations due to their increased architectural complexity and higher number of parameters.

4.2.4 Bar Chart Visualizations

To provide visual clarity on model performance, Figures 4.1, 4.2, and 4.3 present bar charts comparing Accuracy, F1-Score, and AUC Score across the four models.

The visualizations and tabular results indicate that the MobileNetV2 + CBAM (CUBE) model demonstrates improved performance over the baseline MobileNetV2

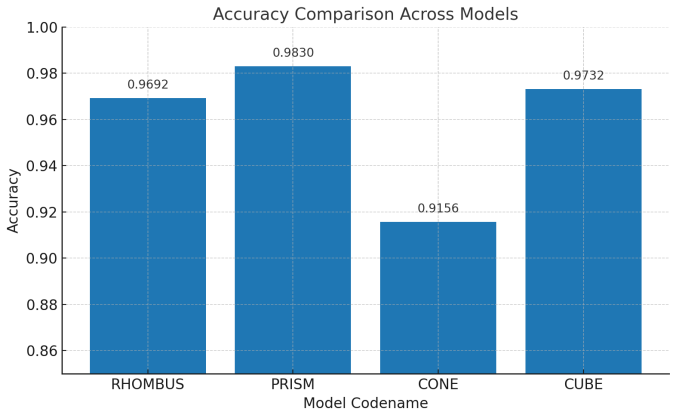

Figure 4.1: Comparison of test accuracy across all models

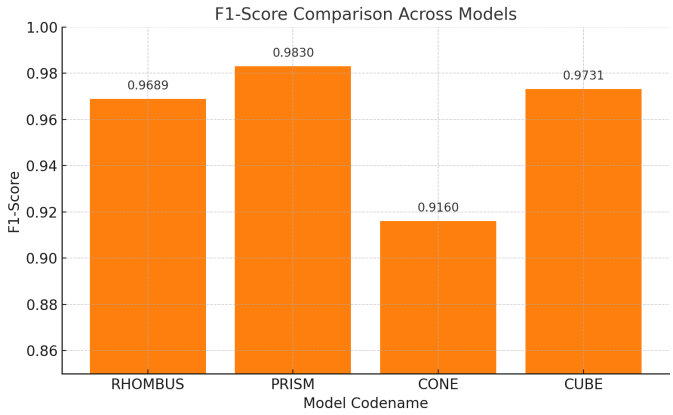

Figure 4.2: Comparison of test F1-score across all models

architecture, with comparable training efficiency. However, while it performs competitively among lightweight models, the Xception model achieves the highest overall performance across evaluation metrics. These findings suggest that attention mechanisms such as CBAM can provide a modest enhancement to feature representation within lightweight architectures.

4.3 Comparative Model Evaluation

This section analyzes and interprets the learning behavior of each model by reviewing the training and validation curves for Accuracy, Loss, and AUC Score. These plots provide insights into the convergence behavior, generalization capability, and

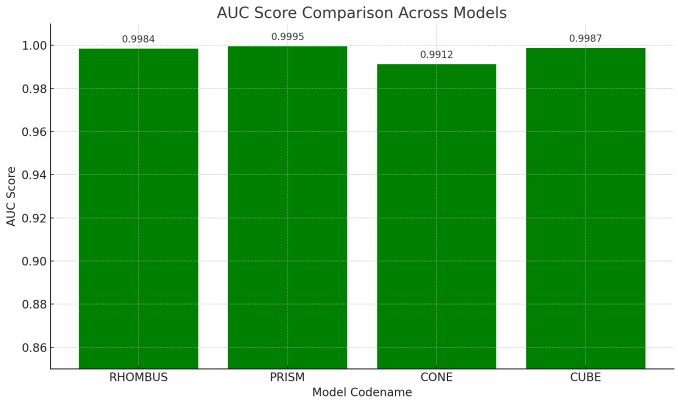

Figure 4.3: Comparison of test AUC score across all models

overfitting tendencies of the models under consistent experimental conditions.

For consistency across models, the reported Precision, Recall, and F1-score values are computed as macro-averaged metrics across all classes. Macro-averaging assigns equal weight to each class, making it particularly suitable for multi-class classification tasks where balanced evaluation across disease categories is required.

All models were trained under a consistent set of hyperparameters to ensure a fair comparison. While this approach enables direct comparison across architectures, it does not account for model-specific hyperparameter tuning, which may further influence performance. A more detailed exploration of hyperparameter sensitivity is left for future work.

4.3.1 RHOMBUS (MobileNetV2)

MobileNetV2 demonstrates a strong upward trend in both training and validation accuracy, with low loss and high AUC score, stabilizing around epoch 20. The model converges quickly, confirming its lightweight yet efficient architecture based on depthwise separable convolutions as shown in Figure 4.4.

4.3.2 PRISM (Xception)

The Xception model shows a steady improvement in AUC and accuracy, although it requires more training time. Its performance plateaued earlier than CUBE, suggesting slightly limited generalization in this setting as shown in Figure 4.5.

4.3.3 CONE (InceptionV3)

While the Inception model required the most training time, its learning curve suggests slower convergence but robust final performance. Despite slight fluctuations,

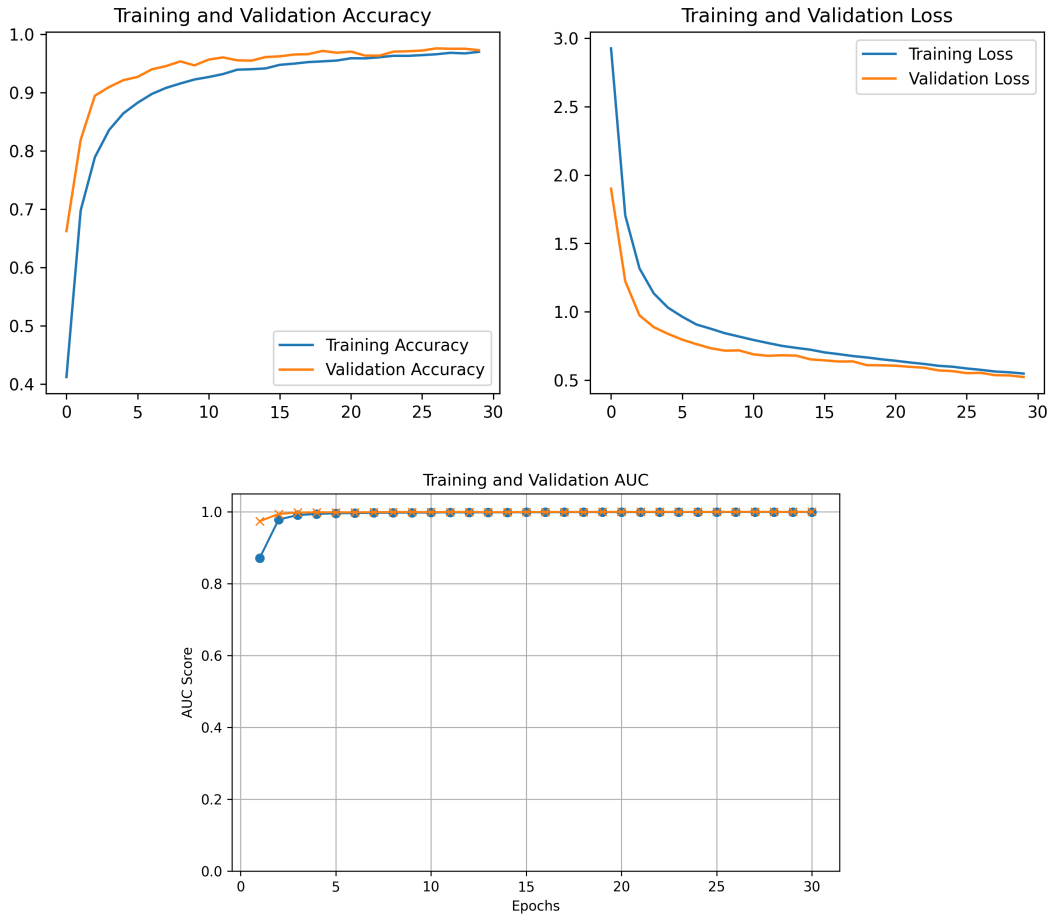

Figure 4.4: RHOMBUS (MobileNetV2): Training vs validation

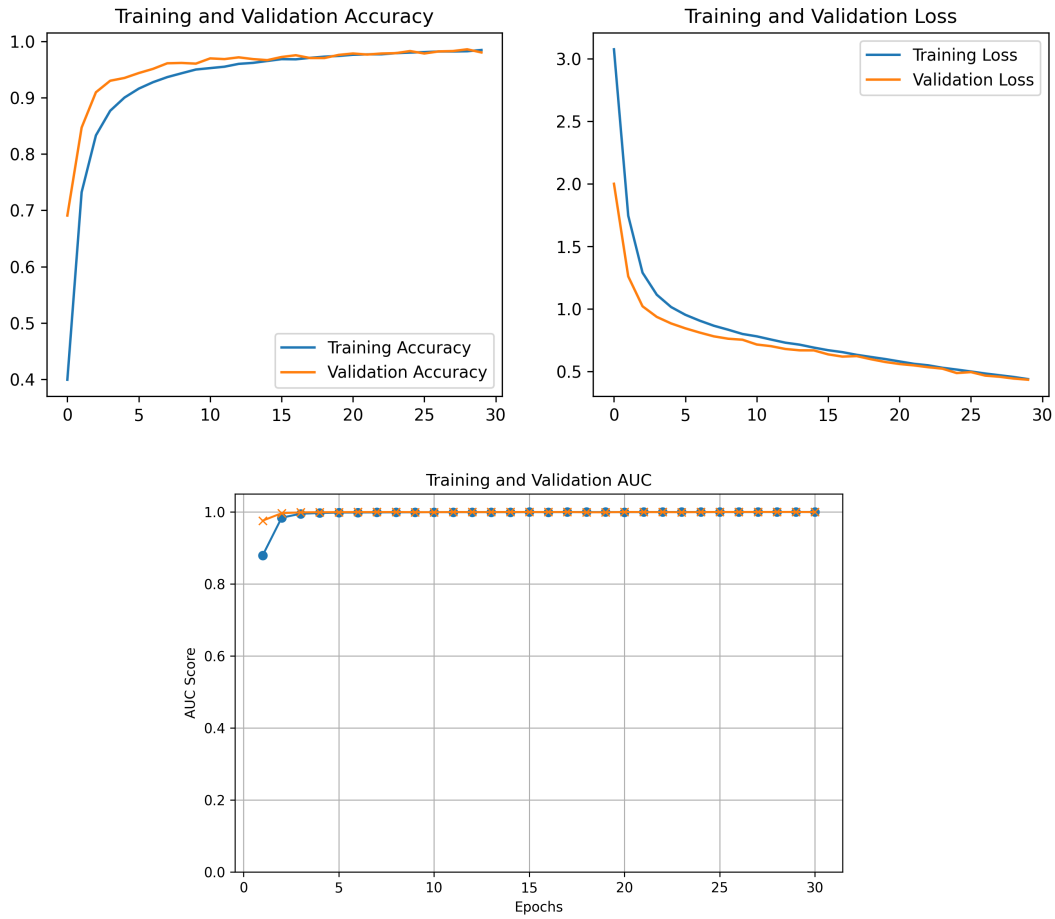

Figure 4.5: PRISM (Xception): Training vs validation

it maintained stable AUC and F1-score values near the end of training as shown in Figure 4.6.

4.3.4 CUBE (MobileNetV2 + CBAM)

The hybrid model incorporating CBAM achieves slightly faster convergence and improved accuracy across all metrics, confirming the effectiveness of attention mechanisms in enhancing feature refinement as shown in Figure 4.7.

4.3.5 Summary

From these comparisons, it is evident that the proposed hybrid model MobileNetV2 + CBAM (CUBE) demonstrates stable convergence behavior and achieves improved performance over the baseline MobileNetV2 model. While the Xception model attains the highest overall classification performance across evaluation metrics, the CUBE model provides a favorable balance between accuracy and computational efficiency. The observed improvements from the integration of CBAM are modest and should be interpreted within the context of lightweight model enhancement.

Furthermore, although the results indicate strong performance on the PlantVillage dataset, which consists of images captured under controlled conditions, the generalization capability of the model to real-world agricultural environments requires further validation. Therefore, the findings suggest that the proposed approach is promising for resource-constrained applications, but its practical deployment should be considered with appropriate limitations.

4.4 Model Per-Class Classification Evaluation

To provide a detailed assessment of model classification performance beyond aggregate metrics, a per-class evaluation was conducted for each of the 38 classes in the PlantVillage dataset. Table 4.5, Table 4.6, Table 4.7 and Table 4.8 presents the class-wise classification performance of the MobileNetV2 (RHOMBUS) model in terms of Accuracy, Precision, and Recall.

4.4.1 MobileNetV2 (RHOMBUS)

The per-class evaluation reveals that the RHOMBUS model achieves consistently high performance across the majority of classes, with many disease categories exhibiting precision and recall values above 0.95. Classes such as *Grape___healthy*, *Apple___Cedar_apple_rust*, and *Orange___Haunglongbing* achieve near-perfect classification performance, indicating strong feature separability for these categories.

However, certain classes demonstrate comparatively lower recall values, suggesting challenges in correctly identifying all instances of those diseases. For example, *Tomato___Early_blight* shows a recall of 0.66, indicating that a significant number

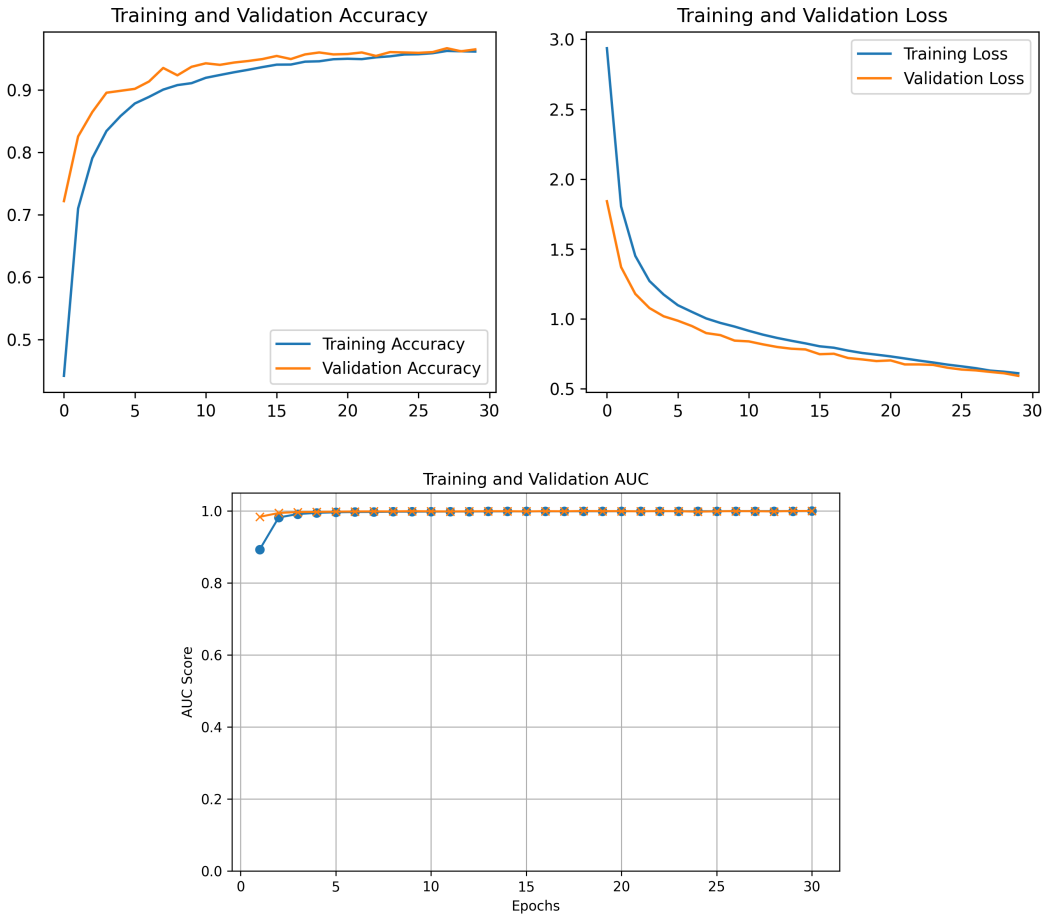

Figure 4.6: CONE (InceptionV3): Training vs validation

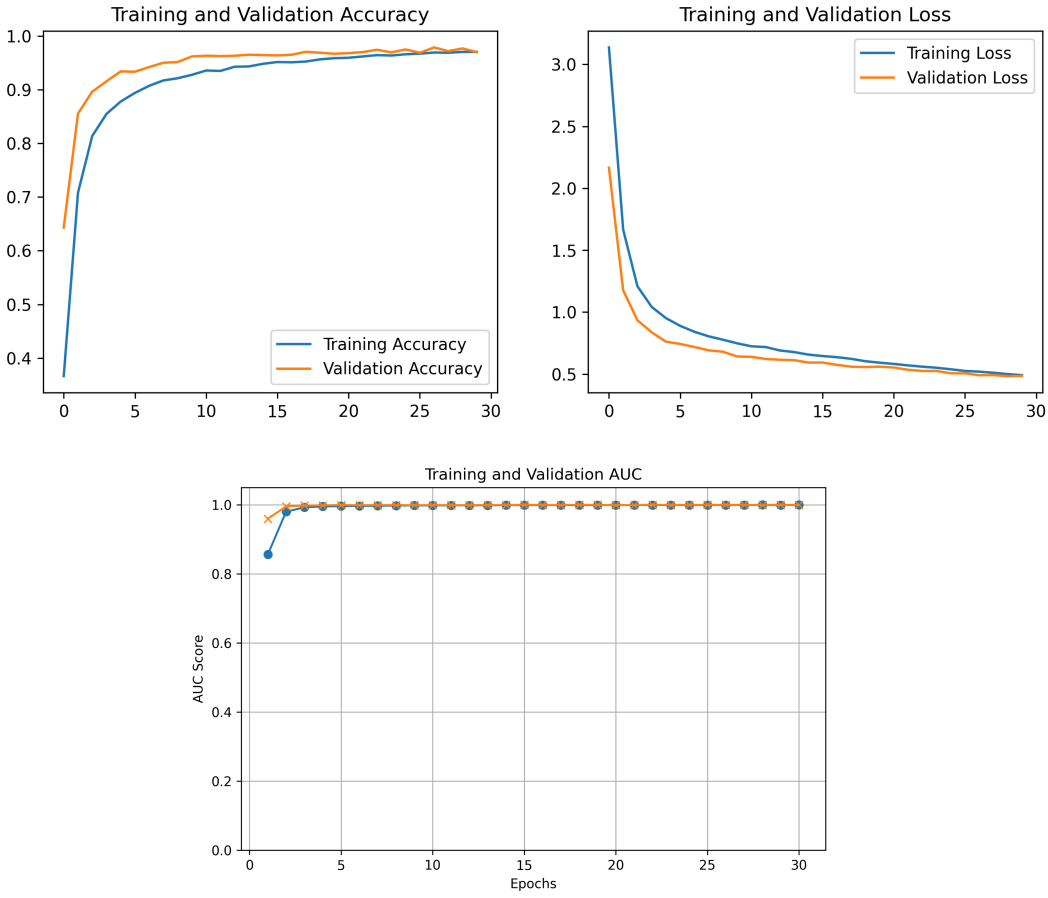

Figure 4.7: CUBE (MobileNetV2 + CBAM): Training vs validation

of samples are misclassified, likely due to visual similarity with other tomato diseases. Similarly, *Corn___Cercospora_leaf_spot* and *Potato___healthy* exhibit reduced recall, reflecting potential overlap in visual characteristics with other classes.

In contrast, some classes display high recall but slightly lower precision, such as *Tomato___Target_Spot*, suggesting that while most true instances are correctly identified, there is some confusion with other classes. Overall, these results indicate that while the model performs strongly across most categories, performance variability exists among visually similar disease classes, highlighting opportunities for further refinement in feature discrimination.

Table 4.5: Per-class classification performance of MobileNetV2 (RHOMBUS)

Class	**Accuracy**	**Precision**	**Recall**
Apple___Apple_scab	0.99	0.91	0.99
Apple___Black_rot	0.99	0.98	0.99
Apple___Cedar_apple_rust	1.00	1.00	1.00
Apple___healthy	0.99	0.98	0.99
Blueberry___healthy	0.99	1.00	0.99
Cherry_(including_sour)___Powdery_mildew	0.97	0.99	0.97
Cherry_(including_sour)___healthy	1.00	1.00	1.00
Corn_(maize)___Cercospora_leaf_spot			
Gray_leaf_spot	0.82	0.89	0.82
Corn_(maize)___Common_rust	1.00	0.98	1.00
Corn_(maize)___Northern_Leaf_Blight	0.92	0.91	0.92
Corn_(maize)___healthy	0.98	1.00	0.98
Grape___Black_rot	0.98	0.98	0.98
Grape___Esca_(Black_Measles)	0.99	0.99	0.99
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	1.00	1.00	1.00
Grape___healthy	1.00	1.00	1.00
Orange___Haunglongbing_(Citrus_greening)	1.00	1.00	1.00
Peach___Bacterial_spot	1.00	0.99	1.00
Peach___healthy	0.96	0.93	0.96
Pepper,_bell___Bacterial_spot	0.99	0.97	0.99
Pepper,_bell___healthy	0.99	0.99	0.99
Potato___Early_blight	1.00	0.94	1.00
Potato___Late_blight	0.93	0.99	0.93
Potato___healthy	0.87	0.95	0.87
Raspberry___healthy	1.00	1.00	1.00
Soybean___healthy	1.00	0.99	1.00
Squash___Powdery_mildew	0.99	1.00	0.99
Strawberry___Leaf_scorch	1.00	0.99	1.00

Continued on next page

Table 4.5 continued from previous page

Class	Accuracy	Precision	Recall
Strawberry___healthy	1.00	1.00	1.00
Tomato___Bacterial_spot	0.93	0.95	0.93
Tomato___Early_blight	0.66	0.95	0.66
Tomato___Late_blight	0.96	0.94	0.96
Tomato___Leaf_Mold	0.81	1.00	0.81
Tomato___Septoria_leaf_spot	0.97	0.88	0.97
Tomato___Spider_mites Two-spotted_spider_mite	0.90	0.96	0.90
Tomato___Target_Spot	0.93	0.79	0.93
Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.98	1.00	0.98
Tomato___Tomato_mosaic_virus	0.93	0.98	0.93
Tomato___healthy	0.99	0.89	0.99

4.4.2 MobileNetV2 + CBAM(CUBE)

The per-class evaluation of the CUBE model indicates consistently strong classification performance across the majority of plant disease categories, with many classes achieving precision and recall values above 0.95. Classes such as *Apple___healthy*, *Peach___Bacterial_spot*, and *Orange___Haunglongbing* demonstrate near-perfect precision and recall, indicating clear feature separability and effective attention-guided feature refinement.

Compared to the baseline MobileNetV2 model, several classes show slight improvements in recall, suggesting that the integration of CBAM enhances the model's ability to correctly identify disease-affected regions. However, performance variability is still observed in certain visually complex classes. For instance, *Tomato___Early_blight* exhibits a relatively low recall of 0.69, indicating difficulty in capturing all true instances, likely due to similarity with other tomato diseases. Similarly,
Corn___Cercospora_leaf_spot and *Potato___healthy* present lower recall values, reflecting overlapping visual patterns with other categories.

In contrast, some classes such as *Tomato___Target_Spot* demonstrate high recall but comparatively lower precision, suggesting that while most true cases are detected, there is some misclassification with similar disease classes. Overall, these results indicate that the CUBE model maintains strong and consistent performance across most categories, with modest improvements over the baseline model, while still facing challenges in distinguishing visually similar disease patterns.

Table 4.6: Per-class classification performance of MobileNetV2 + CBAM (CUBE)

Class	Accuracy	Precision	Recall
Apple___Apple_scab	0.96	0.93	0.96
Apple___Black_rot	1.00	0.98	1.00
Apple___Cedar_apple_rust	0.98	1.00	0.98
Apple___healthy	1.00	0.99	1.00
Blueberry___healthy	0.99	1.00	0.99
Cherry_(including_sour)___Powdery_mildew	0.98	0.99	0.98
Cherry_(including_sour)___healthy	1.00	1.00	1.00
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	0.86	0.89	0.86
Corn_(maize)___Common_rust	1.00	0.96	1.00
Corn_(maize)___Northern_Leaf_Blight	0.91	0.94	0.91
Corn_(maize)___healthy	0.99	1.00	0.99
Grape___Black_rot	0.97	0.99	0.97
Grape___Esca_(Black_Measles)	1.00	0.98	1.00
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	1.00	1.00	1.00
Grape___healthy	1.00	1.00	1.00
Orange___Haunglongbing_(Citrus_greening)	1.00	1.00	1.00
Peach___Bacterial_spot	1.00	1.00	1.00
Peach___healthy	1.00	0.95	1.00
Pepper,_bell___Bacterial_spot	0.95	0.98	0.95
Pepper,_bell___healthy	1.00	0.97	1.00
Potato___Early_blight	1.00	0.94	1.00
Potato___Late_blight	0.93	1.00	0.93
Potato___healthy	0.87	0.95	0.87
Raspberry___healthy	1.00	1.00	1.00
Soybean___healthy	1.00	0.99	1.00
Squash___Powdery_mildew	0.99	1.00	0.99
Strawberry___Leaf_scorch	1.00	0.99	1.00
Strawberry___healthy	1.00	1.00	1.00
Tomato___Bacterial_spot	0.95	0.95	0.95
Tomato___Early_blight	0.69	0.96	0.69
Tomato___Late_blight	0.97	0.95	0.97
Tomato___Leaf_Mold	0.87	0.97	0.87
Tomato___Septoria_leaf_spot	0.95	0.95	0.95
Tomato___Spider_mites Two-spotted_spider_mite	0.94	0.96	0.94
Tomato___Target_Spot	0.96	0.76	0.96

Continued on next page

Table 4.6 continued

Class	Accuracy	Precision	Recall
Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.99	1.00	0.99
Tomato___Tomato_mosaic_virus	0.98	0.96	0.98
Tomato___healthy	0.96	0.95	0.96

4.4.3 Xception(PRISM)

The per-class evaluation of the PRISM (Xception) model demonstrates consistently high classification performance across nearly all plant disease categories, with the majority of classes achieving precision and recall values close to or equal to 1.00. This indicates strong feature representation and effective class discrimination capabilities of the model.

Several classes including *Apple___Black_rot*, *Grape___healthy*, and *Soybean___healthy*, achieve near-perfect precision and recall, reflecting clear separability of these categories. Compared to the lightweight models, PRISM shows improved recall in many classes, suggesting a stronger ability to correctly identify true instances across the dataset.

However, some classes still exhibit relatively lower recall or precision. For example, *Apple___Apple_scab* shows a recall of 0.94, while *Corn___Northern_Leaf_Blight* and *Tomato___Early_blight* demonstrate moderate recall values, indicating occasional misclassification. Additionally, classes such as *Corn___Cercospora_leaf_spot* exhibit lower precision, suggesting confusion with visually similar disease patterns.

Overall, these results indicate that the PRISM model achieves the highest and most consistent per-class performance among the evaluated architectures, although minor variability remains in classes with similar visual characteristics.

Table 4.7: Per-class classification performance of Xception (PRISM)

Class	Accuracy	Precision	Recall
Apple___Apple_scab	0.94	1.00	0.94
Apple___Black_rot	1.00	1.00	1.00
Apple___Cedar_apple_rust	1.00	1.00	1.00
Apple___healthy	1.00	0.97	1.00
Blueberry___healthy	1.00	1.00	1.00
Cherry_(including_sour)___Powdery_mildew	0.99	0.99	0.99
Cherry_(including_sour)___healthy	1.00	1.00	1.00
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	0.91	0.80	0.91
Corn_(maize)___Common_rust	0.99	0.99	0.99

Continued on next page

Table 4.7 continued

Class	Accuracy	Precision	Recall
Corn_(maize)___Northern_Leaf_Blight	0.87	0.94	0.87
Corn_(maize)___healthy	1.00	1.00	1.00
Grape___Black_rot	0.99	0.99	0.99
Grape___Esca_(Black_Measles)	1.00	0.99	1.00
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	1.00	0.99	1.00
Grape___healthy	1.00	1.00	1.00
Orange___Haunglongbing_(Citrus_greening)	1.00	1.00	1.00
Peach___Bacterial_spot	1.00	1.00	1.00
Peach___healthy	1.00	0.98	1.00
Pepper,_bell___Bacterial_spot	1.00	1.00	1.00
Pepper,_bell___healthy	1.00	1.00	1.00
Potato___Early_blight	1.00	0.98	1.00
Potato___Late_blight	0.99	0.98	0.99
Potato___healthy	0.91	1.00	0.91
Raspberry___healthy	1.00	1.00	1.00
Soybean___healthy	1.00	1.00	1.00
Squash___Powdery_mildew	1.00	1.00	1.00
Strawberry___Leaf_scorch	0.99	1.00	0.99
Strawberry___healthy	1.00	1.00	1.00
Tomato___Bacterial_spot	0.98	0.96	0.98
Tomato___Early_blight	0.89	0.89	0.89
Tomato___Late_blight	0.95	0.97	0.95
Tomato___Leaf_Mold	0.91	0.99	0.91
Tomato___Septoria_leaf_spot	0.96	0.96	0.96
Tomato___Spider_mites Two-spotted_spider_mite	0.94	0.97	0.94
Tomato___Target_Spot	0.94	0.90	0.94
Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.99	1.00	0.99
Tomato___Tomato_mosaic_virus	1.00	0.93	1.00
Tomato___healthy	1.00	0.98	1.00

4.4.4 Inception (CONE)

The per-class evaluation of the CONE (Inception) model demonstrates strong overall classification performance, with many classes achieving precision and recall values above 0.95. Classes such as *Blueberry___healthy*, *Grape___healthy*, and *Soybean___healthy* exhibit near-perfect performance, indicating effective feature extraction for clearly distinguishable categories.

However, compared to the PRISM and CUBE models, greater variability is observed in certain disease classes. For instance, *Potato___healthy* shows a relatively low recall of 0.78, indicating difficulty in correctly identifying all healthy samples. Similarly, *Tomato___Early_blight* and *Tomato___Leaf_Mold* exhibit reduced recall values, suggesting confusion with visually similar disease conditions.

In some cases, higher recall is accompanied by lower precision, such as *Corn___Cercospora_leaf_spot*, indicating that while many true instances are detected, misclassification with other classes still occurs. Overall, the results indicate that while the CONE model performs well across most categories, its performance is slightly less consistent compared to higher-performing architectures, particularly in classes with subtle visual differences.

Table 4.8: Per-class classification performance of Inception (CONE)

Class	**Accuracy**	**Precision**	**Recall**
Apple___Apple_scab	0.91	0.97	0.91
Apple___Black_rot	0.99	1.00	0.99
Apple___Cedar_apple_rust	1.00	0.98	1.00
Apple___healthy	0.99	0.97	0.99
Blueberry___healthy	1.00	1.00	1.00
Cherry_(including_sour)___Powdery_mildew	0.99	0.99	0.99
Cherry_(including_sour)___healthy	1.00	1.00	1.00
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	0.95	0.75	0.95
Corn_(maize)___Common_rust	0.99	0.97	0.99
Corn_(maize)___Northern_Leaf_Blight	0.81	0.97	0.81
Corn_(maize)___healthy	0.99	1.00	0.99
Grape___Black_rot	0.97	0.96	0.97
Grape___Esca_(Black_Measles)	0.98	0.98	0.98
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	0.99	0.99	0.99
Grape___healthy	1.00	1.00	1.00
Orange___Haunglongbing_(Citrus_greening)	1.00	1.00	1.00
Peach___Bacterial_spot	0.99	0.99	0.99
Peach___healthy	0.98	0.96	0.98
Pepper,_bell___Bacterial_spot	0.97	0.97	0.97
Pepper,_bell___healthy	0.99	0.99	0.99
Potato___Early_blight	0.99	0.97	0.99
Potato___Late_blight	0.95	0.95	0.95
Potato___healthy	0.78	0.95	0.78
Raspberry___healthy	1.00	0.98	1.00
Soybean___healthy	1.00	1.00	1.00

Continued on next page

Table 4.8 continued

Class	Accuracy	Precision	Recall
Squash___Powdery_mildew	0.99	1.00	0.99
Strawberry___Leaf_scorch	1.00	1.00	1.00
Strawberry___healthy	1.00	1.00	1.00
Tomato___Bacterial_spot	0.93	0.95	0.93
Tomato___Early_blight	0.83	0.87	0.83
Tomato___Late_blight	0.92	0.95	0.92
Tomato___Leaf_Mold	0.83	0.93	0.83
Tomato___Septoria_leaf_spot	0.94	0.89	0.94
Tomato___Spider_mites Two-spotted_spider_mite	0.92	0.95	0.92
Tomato___Target_Spot	0.89	0.86	0.89
Tomato___Tomato_Yellow_Leaf_Curl_Virus	1.00	0.99	1.00
Tomato___Tomato_mosaic_virus	0.95	0.87	0.95
Tomato___healthy	0.97	0.94	0.97

4.5 Grad-CAM Visualizations

To provide interpretability into the decision-making process of the trained deep learning models, Gradient-weighted Class Activation Mapping (Grad-CAM) was employed. Grad-CAM generates visual explanations by highlighting regions of the input leaf image that contribute most to the model's prediction. These visualizations offer qualitative insights into model behavior but should not be interpreted as definitive evidence of model correctness.

4.5.1 Interpretability Through Visual Attention

Grad-CAM visualizations were generated for correctly classified images from each model to examine attention patterns. The activation maps provide an indication of where the model focuses when making predictions, such as whether attention is directed toward disease-affected regions or extends to surrounding areas. However, these observations are illustrative in nature and do not guarantee that the model's reasoning aligns perfectly with the true underlying disease features.

4.5.2 Model-Specific Observations

- **MobileNetV2 (RHOMBUS)**: The RHOMBUS model generally exhibited broader activation patterns across the leaf surface. In some cases, attention appeared to extend beyond the primary lesion regions, suggesting a more diffuse focus.

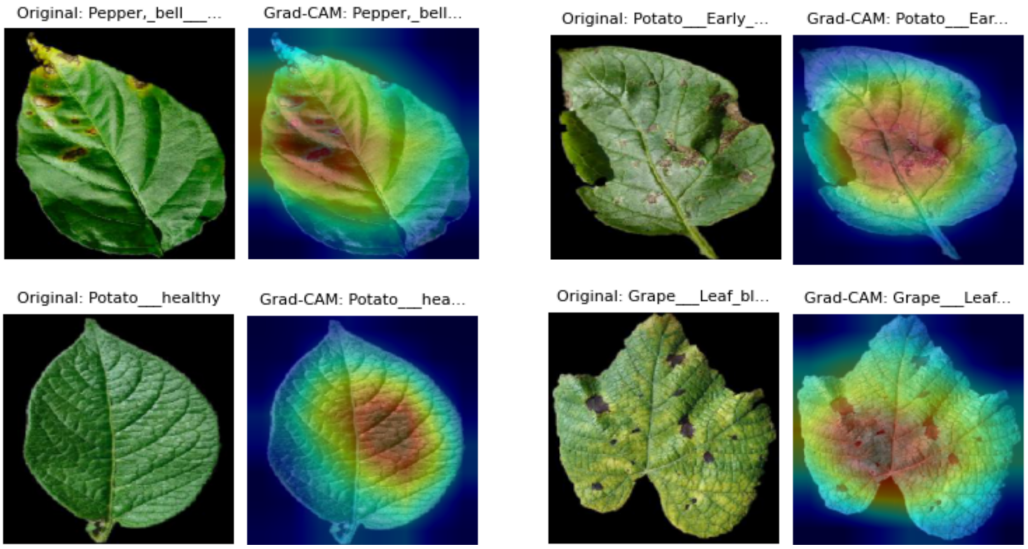

Figure 4.8: Grad-CAM visualization comparison for example leaf images across RHOMBUS, PRISM, CONE, and CUBE

- **MobileNetV2 + CBAM (CUBE)**: The CUBE model often produced more localized activation patterns compared to the baseline MobileNetV2 model. The integration of CBAM appears to influence the spatial distribution of attention; however, these visualizations should be interpreted as qualitative indicators rather than conclusive evidence of improved model reasoning.
- **Xception (PRISM)**: The PRISM model demonstrated a combination of localized and distributed attention patterns. Grad-CAM maps frequently highlighted lesion-centered regions, though some activation in non-diseased areas was also observed.
- **Inception (CONE)**: The CONE model showed relatively focused activation in certain cases, with attention appearing concentrated around lesion regions. However, as with other models, these observations are qualitative and may vary across samples.

4.5.3 Sample Visualizations

Figure 4.8 presents sample Grad-CAM outputs across the four models for selected input images. While differences in attention patterns can be observed between models, these visualizations are intended to provide illustrative examples of model behavior rather than definitive comparisons of performance or correctness.

Table 4.9: Model performance metrics on test set

Model	Accuracy	Precision	Recall	F1-Score	AUC
Inception (CONE)	0.9156	0.9225	0.9156	0.9160	0.9912
MobileNetV2 (RHOMBUS)	0.9692	0.9707	0.9692	0.9689	0.9984
MobileNetV2 + CBAM (CUBE)	0.9732	0.9747	0.9732	0.9731	0.9987
Xception (PRISM)	**0.9830**	**0.9833**	**0.9830**	**0.9830**	**0.9995**

4.6 Statistical Discussion & Observations

To critically assess the performance of each model, a comprehensive statistical comparison was conducted across five key metrics: Accuracy, Precision, Recall, F1-Score, and AUC. Table 4.9 summarizes the evaluation results obtained from the test set for all four models.

4.6.1 Performance Observations

- **Xception (PRISM)** demonstrated the highest performance across all evaluation metrics, achieving 98.30% accuracy and the highest AUC score (0.9995), suggesting strong discriminatory power and generalization capability.
- **MobileNetV2 + CBAM (CUBE)** closely followed with high precision (0.9747) and an AUC score of 0.9987, indicating that the integration of attention mechanisms improves model sensitivity and focus on disease regions.
- **MobileNetV2 (RHOMBUS)** maintained a strong balance between accuracy and computational efficiency, validating its architectural strength using Depthwise Separable Convolutions (DSC).
- **Inception (CONE)**, although achieving respectable results, trailed behind in all metrics, particularly with the lowest accuracy (91.56%) and F1-score (91.60%), which may be attributed to its heavier architecture and longer inference time.

4.6.2 Trade-Offs and Implications

While Xception emerged as the statistically superior model, the marginal gains in accuracy over the MobileNet-based models come at the cost of increased computational complexity and training time. On the other hand, MobileNetV2 and its CBAM-enhanced variant provided a near-optimal balance between performance

Table 4.10: Quantized model deployment metrics on mobile device

Model	Model Size (MB)	Avg. Inference Time (ms)
RHOMBUS (MobileNetV2)	3.7	156.0
CUBE (MobileNetV2 + CBAM)	4.1	186.6
PRISM (Xception)	22.5	291.7
CONE (Inception)	23.0	334.9

and efficiency, making them well-suited for deployment on edge devices and mobile applications.

Therefore, in environments constrained by power, memory, or latency (e.g., farms or rural areas), **MobileNetV2 + CBAM** is preferred, while Xception is ideal in cloud-based or high-performance inference scenarios.

4.7 Deployment Performance on Mobile

This section evaluates the deployment performance of the trained models when converted to a mobile-compatible format. In addition to inference speed and model size, this analysis considers the effects of post-training optimization and the practical constraints of edge-device environments.

4.7.1 Deployment Setup

All trained models were converted to TensorFlow Lite (`.tflite`) format using post-training dynamic range quantization. Deployment testing was carried out on an Android device equipped with a Snapdragon processor and 6GB RAM, representing a typical mid-range mobile platform.

4.7.2 Performance Metrics

Inference times were benchmarked over multiple test samples, and average latency per image was recorded. Table 4.10 summarizes the model size and inference performance for each model after quantization.

4.7.3 Post-Quantization Observations

The application of dynamic range quantization significantly reduced model sizes while maintaining functional inference capability on mobile hardware. Although classification accuracy after quantization was not extensively re-evaluated in this study, prior research suggests that dynamic range quantization typically results in minimal accuracy degradation. A more systematic evaluation of post-quantization accuracy remains an important direction for future work.

4.7.4 Discussion and Trade-Offs

The results indicate that MobileNet-based models, particularly RHOMBUS and CUBE, are more suitable for mobile deployment due to their smaller model sizes and lower inference latency. While PRISM (Xception) achieves higher classification accuracy in the benchmark setting, its increased computational cost and memory requirements limit its practicality in resource-constrained environments.

The CUBE model introduces a moderate increase in inference time compared to the baseline MobileNetV2 model, reflecting the additional computational overhead of the CBAM module. However, this increase remains within acceptable bounds for real-time inference on mobile devices.

It is important to note that deployment performance may vary across different hardware configurations, operating systems, and inference engines. Therefore, the reported results should be interpreted as indicative rather than universally representative. A more comprehensive deployment analysis across multiple device types and conditions would provide a deeper understanding of real-world performance variability.

Overall, the findings suggest that lightweight architectures offer a practical balance between performance and efficiency for mobile-based plant disease detection, although further validation is required to assess robustness across diverse deployment environments.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This research presented a comprehensive approach to the detection and classification of plant diseases using hybrid deep learning models. By leveraging the strengths of convolutional neural networks (CNNs) and attention mechanisms such as the Convolutional Block Attention Module (CBAM), the proposed approach aimed to improve classification performance while maintaining computational efficiency suitable for resource-constrained environments.

Among the evaluated architectures — MobileNetV2 (RHOMBUS), MobileNetV2 + CBAM (CUBE), Xception (PRISM), and Inception (CONE) — the results summarized in Table 4.1 indicate that the Xception model achieved the highest overall classification performance across evaluation metrics. The MobileNetV2 + CBAM (CUBE) model, however, demonstrated a favorable balance between accuracy and computational efficiency, achieving an AUC score of 0.9987 and an accuracy of 97.32% while maintaining a relatively small model size and efficient inference time. The integration of the CBAM module provided a modest improvement over the baseline MobileNetV2 model within the class of lightweight architectures, rather than outperforming higher-capacity models such as Xception. Additionally, Grad-CAM visualizations provided qualitative insights into the regions of interest learned by the model, but these should be interpreted as illustrative rather than definitive evidence of model correctness.

This work demonstrates that lightweight models such as MobileNetV2, when enhanced with attention mechanisms and efficient convolutional operations, can achieve strong performance on benchmark datasets such as PlantVillage. However, since the evaluation is conducted on images captured under controlled conditions, the results should be interpreted within the context of benchmark performance. While preliminary deployment experiments using TensorFlow Lite on Android devices indicate the feasibility of on-device inference, further validation on real-world field data is required to fully assess generalization and practical deployment

readiness in agricultural environments.

5.2 Research Contributions

This research contributes to the development of efficient deep learning approaches for automated plant disease detection. Specifically, the study proposes a hybrid deep learning framework that integrates lightweight convolutional neural network architecture MobileNetV2 with the Convolutional Block Attention Module (CBAM) to enhance feature extraction and improve classification performance. The research also investigates the effectiveness of transfer learning and attention-based feature refinement for identifying subtle plant disease patterns using the PlantVillage Dataset. Furthermore, the study provides a comparative evaluation of the proposed hybrid model against baseline deep learning models in terms of classification accuracy and computational efficiency. The findings contribute to the growing field of precision agriculture by demonstrating the potential of lightweight, attention-enhanced deep learning models for practical plant disease detection systems.

5.3 Future Work

Future research may focus on extending the proposed framework to handle more complex and realistic agricultural environments. One potential direction is the incorporation of real-world leaf images captured under diverse field conditions, including variations in lighting, background clutter, noise, and occlusion. Such data would enable the development of models that are more robust and better suited for practical deployment in real farming scenarios. Future studies may also explore the development of multi-disease detection models capable of identifying multiple infections within a single plant sample. Since crops are often affected by more than one disease simultaneously, such models would provide a more comprehensive diagnostic tool for farmers and agricultural practitioners. Furthermore, the application of federated learning techniques could be investigated to enable collaborative model training across distributed agricultural datasets while preserving the privacy of farmer data. Finally, future work may focus on implementing a complete end-to-end mobile or edge-based system with an intuitive user interface and offline capabilities, allowing farmers to perform real-time disease diagnosis directly in the field.

Bibliography

[1] G. N. Agrios, *Plant Pathology*, Academic Press, 5th ed., 2005.

[2] B. R. Barbedo, *Digital image processing techniques for detecting, quantifying and classifying plant diseases*, SpringerPlus, 2 (2013), pp. 1–12.

[3] F. Chollet, *Xception: Deep learning with depthwise separable convolutions*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1251–1258.

[4] A. G. H. et al., *Mobilenets: Efficient convolutional neural networks for mobile vision applications*, arXiv preprint, (2017).

[5] B. H. et al., *Genome sequence and analysis of the irish potato famine pathogen phytophthora infestans*, Nature, 461 (2009), pp. 393–398.

[6] C. S. et al., *Rethinking the inception architecture for computer vision*, in Proc. CVPR, 2016, pp. 2818–2826.

[7] F. S. et al., *The global burden of pathogens and pests on major food crops*, Nature Ecology & Evolution, 3 (2019), pp. 430–439.

[8] J. A. C. et al., *Machine learning applications in plant disease classification: A review*, Computers and Electronics in Agriculture, 145 (2018), pp. 272–285.

[9] J. B. R. et al., *The persistent threat of emerging plant disease pandemics*, PLOS Pathogens, 17 (2021), p. e1009584.

[10] J. D. et al., *The top 10 fungal pathogens in molecular plant pathology*, Molecular Plant Pathology, 13 (2012), pp. 414–430.

[11] J. M. et al., *Recent advances in sensing plant diseases for precision crop protection*, European Journal of Plant Pathology, 133 (2012), pp. 197–209.

[12] J. Z. et al., *Applications of computer vision in plant pathology*, Journal of Integrative Agriculture, 13 (2014), pp. 1537–1551.

[13] M. M. et al., *Top 10 bacterial pathogens in molecular plant pathology*, Molecular Plant Pathology, 13 (2012), pp. 614–629.

[14] R. A. N. et al., *Advances in virus detection in plants*, Crop Science, 56 (2016), pp. 1232–1246.

[15] S. S. et al., *Deep neural networks based recognition of plant diseases by leaf image classification*, Computational Intelligence and Neuroscience, 2016 (2016), pp. 1–11.

[16] K. P. Ferentinos, *Deep learning models for plant disease detection and diagnosis*, Computers and Electronics in Agriculture, 145 (2018), pp. 311–318.

[17] Food and Agriculture Organization, *Protecting plant health is essential*. FAO Report, May 2022. Accessed: Aug. 24, 2025.

[18] B. D. Harrison and A. F. Murant, *Plant viruses and virus diseases: Progress and problems*, Philosophical Transactions of the Royal Society B: Biological Sciences, 321 (1988), pp. 457–471.

[19] J. Hu, L. Shen, and G. Sun, *Squeeze-and-excitation networks*, in Proc. CVPR, 2018, pp. 7132–7141.

[20] D. P. Hughes and M. Salathé, *An open access repository of images on plant health to enable the development of mobile disease diagnostics*, arXiv preprint arXiv:1511.08060, (2015).

[21] A. Kamilaris and F. Prenafeta-Boldú, *Deep learning in agriculture: A survey*, Computers and Electronics in Agriculture, 147 (2018), pp. 70–90.

[22] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014).

[23] J. Liang, Y. Ma, and X. Li, *Applications of ai in agriculture: A review*, Precision Agriculture, 21 (2020), pp. 267–291.

[24] J. Loh, D. Singh, M. Ghosh, and A. Sahoo, *Hyperspectral imaging for early plant disease detection: A review*, Journal of Imaging, 8 (2022), p. 130.

[25] J. Mahlein, *Plant disease detection by imaging sensors – parallels and specific demands for precision agriculture*, Computers and Electronics in Agriculture, 100 (2014), pp. 139–149.

[26] H. Marschner, *Marschner's Mineral Nutrition of Higher Plants*, Elsevier, 3rd ed., 2012.

[27] I. D. Mienye, T. G. Swart, G. Obaido, M. Jordan, and P. Ilono, *Deep convolutional neural networks in medical image analysis: A review*, Information, 16 (2025), p. 195.

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

[29] C. Ozdemir, Y. Dogan, and Y. Kaya, *A new local pooling approach for convolutional neural network: Local binary pattern*, Multimedia Tools and Applications, 83 (2024), pp. 34137–34151.

[30] R. Patil, A. Pawar, and A. Kolekar, *Hybrid cnn model for leaf disease classification*, Multimedia Tools and Applications, 81 (2022), pp. 9597–9613.

[31] R. N. Perry and M. Moens, *Plant Nematology*, CABI Publishing, 2006.

[32] A. Ramcharan, K. Baranowski, P. McCloskey, B. Ahmed, J. Legg, and D. P. Hughes, *A mobile-based deep learning model for cassava disease diagnosis*, Frontiers in Plant Science, 10 (2019), p. 272.

[33] A. Ramcharan, K. Baranowski, P. McCloskey, B. A. Ahmed, J. Legg, and D. P. Hughes, *A mobile-based deep learning model for cassava disease diagnosis*, Frontiers in Plant Science, 10 (2019), p. 272.

[34] Saiwa Inc., *Image Segmentation: Techniques & Types*, 2025. Accessed: Aug. 23, 2025.

[35] ——, *Plant Disease Identification and Control: A Complete Guide*, 2025. Accessed: Aug. 22, 2025.

[36] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, *Mobilenetv2: Inverted residuals and linear bottlenecks*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, June 2018, pp. 4510–4520.

[37] S. Sankaran, A. Mishra, R. Ehsani, and C. Davis, *A review of advanced techniques for detecting plant diseases*, Computers and Electronics in Agriculture, 72 (2010), pp. 1–13.

[38] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, *Grad-cam: Visual explanations from deep networks via gradient-based localization*, in IEEE International Conference on Computer Vision (ICCV), 2017, pp. 618–626.

[39] T. Shorten and T. M. Khoshgoftaar, *A survey on image data augmentation for deep learning*, Journal of Big Data, 6 (2019).

[40] D. Singh, N. Jain, P. Jain, P. Kayal, S. Kumawat, and N. Batra, *Plantdoc: A dataset for visual plant disease detection*, in Proceedings of the 2020 ACM on Multimedia Conference, 2020, pp. 2496–2505.

[41] P. Strange and P. Scott, *Plant disease: A threat to global food security*, Annual Review of Phytopathology, 43 (2005), pp. 83–116.

[42] M. Tan and Q. V. Le, *Efficientnet: Rethinking model scaling for convolutional neural networks*, in International Conference on Machine Learning (ICML), 2019, pp. 6105–6114.

[43] U.S. Department of Agriculture, *UN Declares 2020 International Year of Plant Health*, Jan. 2020. USDA News. Accessed: Aug. 24, 2025.

[44] S. Woo, J. Park, J. Lee, and I. S. Kweon, *Cbam: Convolutional block attention module*, in Proc. ECCV, 2018, pp. 3–19.

[45] Y. Zhang, H. Jiang, and X. Liu, *Attention mechanisms in deep learning: A review on applications to agricultural vision*, Computers and Electronics in Agriculture, 198 (2022), p. 107120.

[46] C. Zhou, Y. Zhou, and J. Zhang, *Hybrid deep learning framework for plant disease classification*, Computers and Electronics in Agriculture, 190 (2021), p. 106481.