NATIONAL ECONOMICS UNIVERSITY

FACULTY OF ECONOMICAL MATHEMATICS

GRADUATION THESIS

✧ ✧ ✧

Plant Disease Detection with A Transfer Learning Approach

✧ ✧ ✧

Student	:	**Hoang Dieu Linh**
Student ID	:	**11202127**
Class	:	**Data Science in Economics & Business 62**
Instructor	:	**MSc. Nguyen Thanh Tuan**

Ha Noi, May 2024

Acknowledgement

Completing this master's thesis has been an enriching journey, and I would like to take this opportunity to express my sincere gratitude to all those who have supported me along the way.

Foremost, I am deeply indebted to my esteemed supervisor, MSc. Nguyen Thanh Tuan. His guidance throughout this research endeavor has been invaluable. Beyond providing insightful feedback and constructive criticism, he fostered a collaborative environment that empowered me to independently explore research avenues. At the same time, his willingness to offer crucial direction when needed proved instrumental in shaping this thesis. I am particularly grateful for his unwavering support and dedication to my academic growth. The positive impact he has had on my development as a researcher will undoubtedly benefit me throughout my career.

Furthermore, I would like to extend my gratitude to the exceptional faculty members of the Faculty of Mathematical Economics. Their passion for their subjects was truly contagious, and their willingness to engage in stimulating discussions not only broadened my understanding of the field but also motivated me to delve deeper into the subject matter. These stimulating discussions proved to be invaluable stepping stones in the development of my research.

Finally, a sincere thank you to everyone who has contributed to this project, regardless of the scale of their involvement. Your help, inspiration, and contributions, no matter how big or small, have been invaluable on this journey. I am truly grateful to have you as part of my academic experience.

Contents

List of Figures

List of Tables

1 Introduction

Research topic
"Plant Disease Detection with A Transfer Learning Approach"

1.1 Background and Motivation

1.1.1 Importance of Healthy Plants for Agriculture and Environment

Plants are the cornerstone of life on Earth, playing a vital role in both global food security and environmental well-being. In agriculture, healthy plants form the foundation of the food chain. They provide the essential source of calories and nutrients for humans and animals alike. According to the Food and Agriculture Organization (FAO), plant production accounts for over 90% of the world's total food supply.

However, this vital food source is constantly under threat from a variety of pests, including diseases, insects, and weeds. These pests can significantly reduce crop yields. The FAO estimates that globally, all pests combined, including plant diseases, insects, and weeds, account for annual losses of agricultural products between 20-40%. In Vietnam, specifically, pests pose a significant challenge. Plant diseases are a major concern, but infestations by insects and weeds also contribute to substantial crop losses. For instance, a 2018 study published in the Vietnam Journal of Agricultural Sciences estimated that rice blast, a fungal disease, alone causes annual yield losses ranging from 10-30% in Vietnam, depending on the region and rice variety. These losses translate to economic hardship for farmers and threaten food security for the nation's growing population.

Beyond agriculture, healthy plants are crucial for maintaining a healthy environment. Through the process of photosynthesis, plants absorb carbon dioxide, a major greenhouse gas contributing to climate change. They then release oxygen, vital for respiration by humans and animals. Healthy plant communities also play a critical role in soil health. Their root systems help prevent erosion, and their decaying matter enriches the soil, fostering a diverse ecosystem.

In conclusion, healthy plants are essential for both a secure food supply and a healthy environment. Pests of all kinds, including plant diseases, insects, and weeds, pose a significant threat to both of these aspects, causing economic losses and environmental damage. Addressing this challenge by developing efficient and accurate methods for plant disease detection, alongside other pest management strategies, is crucial for ensuring a sustainable future.

1.1.2 Challenges in Traditional Plant Disease Detection Methods

A common approach involves visual inspection by human experts. However, this method suffers from inherent subjectivity. Trained personnel rely on their knowledge and experience to identify disease symptoms based on visual cues like discoloration, lesions, or deformations on leaves. Subtle variations in these symptoms or environmental factors like lighting can lead to misdiagnosis. Furthermore, accurately identifying specific diseases often requires extensive knowledge of plant health and the distinct characteristics of various plant diseases. This level of expertise may not be readily available in all regions, especially in resource-limited areas with limited access to trained agricultural personnel. The lack of readily available expertise creates a barrier to widespread implementation of this method.

Traditional methods, particularly those involving laboratory analysis, can be quite slow and require significant labor. Visual inspection of large fields or numerous samples from different locations can be incredibly time-consuming, especially for large-scale agricultural operations. Additionally, laboratory analysis techniques, such as microscopy or isolation of pathogens, can involve lengthy procedures. These delays in detection can have severe consequences. Diseases can spread rapidly, causing significant damage to crops before they are identified using traditional methods.

Traditional methods often struggle to scale efficiently for large-scale agricultural practices. Manually inspecting vast fields or analyzing numerous samples from various locations can be impractical and require significant manpower. This lack of scalability makes these methods unsuitable for monitoring large plantations or ensuring consistent disease detection across extensive agricultural regions. While not always applicable to all research areas, it's important to acknowledge that even non-visual detection methods may have limitations. These methods, such as those relying on specific instruments or chemical tests, may require specialized equipment or expertise, limiting their accessibility in certain settings.

In conclusion, traditional plant disease detection methods, despite their historical role, present several challenges. Their subjectivity, time-consuming nature, limitations in scalability and early detection, and potential dependence on specialized equipment hinder their effectiveness in modern, large-scale agriculture. These limitations highlight the need for innovative and efficient methods for accurate and timely plant disease detection.

1.1.3 Introduction of deep learning for plant disease detection

The limitations of traditional plant disease detection methods pave the way for the exploration of more advanced solutions. Deep learning, a subfield of artificial intelligence, offers immense potential in this domain. Deep learning algorithms are inspired by the structure and function of the human brain. They consist of artificial neural networks with multiple layers, allowing them to learn complex patterns from vast amounts of data. By analyzing large datasets of labeled images containing healthy and diseased plant leaves, these networks can learn to identify subtle visual cues associated with specific plant diseases.

Deep learning offers a compelling solution for automating plant disease detection processes. Unlike traditional methods that rely on human expertise and visual inspection, deep learning models can analyze images autonomously. This automation eliminates the subjectivity inherent in human decision-making and streamlines the detection process, leading to faster identification of diseases. Deep learning models have the potential to achieve superior accuracy compared to traditional methods. Their ability to learn intricate patterns from vast datasets allows them to differentiate between healthy and diseased leaves with high precision. Furthermore, their automated nature ensures efficiency, enabling the rapid analysis of large volumes of images.

In conclusion, deep learning presents a transformative approach to plant disease detection. Its ability to learn complex patterns, automate processes, and potentially achieve high accuracy offers significant advantages over traditional methods.

1.2 Problem statements

Plant diseases pose a significant threat to global food security and economic stability, particularly in developing countries like Vietnam. These diseases, caused by pathogens such as bacteria, fungi, and viruses, can significantly reduce crop yields and lead to substantial economic losses for farmers. Early and accurate detection of plant diseases is crucial for implementing effective control measures and minimizing damage. Traditional methods for plant disease detection, while offering a baseline for identifying problematic crops, present several limitations. They often rely on visual inspection by human experts, which can be subjective, time-consuming, and require extensive expertise that may not be readily available in all regions. Additionally, laboratory analysis techniques can be slow and impractical for large-scale monitoring. These limitations hinder the effectiveness of traditional methods in ensuring timely and accurate detection of plant diseases in modern agricultural settings.

The limitations of traditional methods necessitate the exploration of more advanced solutions for plant disease detection in Vietnam. There is a pressing need for a method that offers the following advantages:

- Increased Accuracy: Traditional methods may struggle to differentiate between subtle variations in disease symptoms or healthy and diseased leaves, leading to misdiagnosis. A more accurate detection system is needed to minimize false positives and negatives.
- Enhanced Efficiency: Timely detection is crucial for effective disease management. Traditional methods can be slow and labor-intensive, hindering early intervention. A faster and more efficient detection system is required for rapid disease identification.

- Reduced Reliance on Expertise: The dependence on trained personnel in traditional methods limits their scalability and accessibility in resource-limited regions. A solution requiring minimal human intervention is essential.
- Scalability for Large-scale Agriculture: Traditional methods are often unsuitable for monitoring vast agricultural areas. A scalable solution is needed to ensure consistent and reliable detection across large plantations.

This research proposes the application of deep learning for automated and accurate plant disease detection in the Vietnamese agricultural sector. Deep learning algorithms have demonstrated remarkable potential in image classification tasks, including plant disease detection. By leveraging the power of deep learning, this research aims to address the limitations of traditional methods and develop a more efficient, accurate, and scalable solution for plant disease detection in Vietnam. This, in turn, can contribute to improved crop health, increased food security, and enhanced economic stability for Vietnamese farmers.

1.3 Objectives

The world of agriculture is one that is constantly evolving, with new technologies and methodologies being introduced to improve the efficiency and effectiveness of farming practices. One such area that has seen significant advancements in recent years is the detection of plant leaf diseases. This paper aims to delve into this topic, with a specific focus on the application of deep learning, particularly EfficientNet, and the use of Gradient-weighted Class Activation Mapping (Grad-CAM) for model attention visualization.

The first objective of this research is to evaluate the performance of EfficientNet in detecting various plant leaf diseases. EfficientNet, a state-of-the-art convolutional neural network, has shown promising results in various fields, and this research aims to explore its potential in the realm of plant disease detection. This will be achieved by implementing the EfficientNet model, training it on a dataset of plant leaf images, and assessing its performance using various metrics such as accuracy, precision, recall, and F1-score.

In addition to evaluating EfficientNet, this research also aims to utilize Gradient-weighted Class Activation Mapping (Grad-CAM) to visualize the areas of focus of the model when making predictions. By visualizing what the model is focusing on, we can gain valuable insights into the decision-making process of the model. This can help identify potential areas for improvement and contribute to the development of more accurate and efficient models.

Another key objective of this research is to compare the performance of the EfficientNet model with traditional methods of plant disease detection. By comparing the performance of EfficientNet with traditional methods, we can establish the advantages and potential limitations of using deep learning for plant disease detection. This comparison will provide valuable insights that can guide future research and development in this field.

In conclusion, this research aims to explore the application of EfficientNet and Grad-CAM in plant leaf disease detection, compare the performance of EfficientNet with traditional methods, and ultimately optimize the model for better performance. This optimization could involve tuning hyperparameters, experimenting with different data augmentation techniques, or exploring strategies to handle imbalanced data, all with the goal of improving the model's performance and making it more suitable for practical applications. By achieving these objectives, this research hopes to make a significant contribution to the field of plant disease detection and pave the way for future research in this area.

1.4 Scope and Limitations

The scope of this research is primarily centered on the use of EfficientNet and Grad-CAM for detecting various plant leaf diseases. The diseases under consideration are selected based on their prevalence in the agricultural sector and the availability of data. The dataset used in this research consists of a collection of plant leaf images, the size of which is determined by the availability of high-quality images for each of the selected diseases. While other deep learning models may be considered for comparative purposes, the primary focus of this research is on the use of EfficientNet.

However, like any research, this study is not without its limitations. One of the main constraints is the computational resources required for training deep learning models. Despite efforts to optimize the model and the training process, the results may be limited by the available computational resources. Another limitation is the quality and quantity of the data used for training. The accuracy of deep learning models is largely dependent on the quality and diversity of the training data. While every effort will be made to use a comprehensive and high-quality dataset, there may be limitations in terms of the diversity and representation of plant leaf diseases. Lastly, while Grad-CAM will be used to visualize the model's attention, it should be noted that deep learning models are often considered as "black boxes" due to their complex nature. Therefore, the interpretations made from the visualizations should be considered with caution.

Despite these limitations, this research aims to provide valuable insights into the application of EfficientNet for plant leaf disease detection. By clearly defining the scope and acknowledging the limitations, this research hopes to contribute to the field of plant disease detection and pave the way for future research in this area.

1.5 Thesis Organization

This thesis is structured to provide a comprehensive and systematic exploration of the application of machine learning and deep learning techniques to the critical challenge of plant disease detection. The narrative unfolds across six chapters, each serving a distinct purpose in building a cohesive and informative research discourse.

- Chapter 1: Introduction: This chapter sets the stage for the research by providing background information, stating the problem, outlining the objectives, and defining the scope and limitations of the study.

- Chapter 2: Literature Review: This chapter presents a comprehensive review of existing literature pertaining to plant disease detection, highlighting key advancements, challenges, and research gaps in the field. It serves as a foundation for understanding the current state of the art and identifying potential areas for contribution.

- Chapter 3: Theoretical Basis: This chapter establishes the theoretical foundation upon which the research is built and details the technical tools and metrics employed. It delves into the fundamental concepts and principles underlying machine learning, deep learning, and computer vision, with a specific focus on their relevance to plant disease detection. Additionally, it outlines the programming language, frameworks, libraries, and evaluation metrics utilized in this study.

- Chapter 4: Methodology: This chapter describes the research methodology in detail. It includes information about the dataset, preprocessing techniques, model selection process, the implementation of the proposed model and the visualization with Grad-CAM.

- Chapter 5: Experimental Results: This chapter presents the results of the experiments and provides a detailed discussion on the findings. It includes sections on performance comparison of various models, evaluation of the proposed model, and implications of the results.

- Chapter 6: Conclusion and Future Work: This chapter concludes the thesis by summarizing the findings, highlighting the contributions of the study, discussing potential applications, recommending areas for future research, and restating the importance of the research.

2 Literature Review

The application of Machine Learning (ML) and Deep Learning (DL) to plant disease detection has garnered significant attention in recent research. Several studies have explored various approaches to leverage these techniques for accurate and efficient identification of plant diseases and pests.

Early research focused on classifying plant diseases directly from original images, with notable success using deep Convolutional Neural Networks (CNNs) and transfer learning. For instance, Thenmozhi et al. achieved accuracy rates exceeding 95% on insect datasets, while Fang et al. reported 95.61% accuracy using ResNet50 with a modified loss function.

Other studies have investigated classification after locating regions of interest (ROIs), aiming to improve accuracy by focusing on specific areas within images. Nagasubramanian et al. demonstrated the effectiveness of this approach, achieving 95.73% accuracy in identifying soybean stem rot using a 3D DCNN and saliency maps.

As the complexity of plant disease detection tasks increased, researchers began to tackle multi-category classification, where more than two classes of diseases and pests need to be identified. Picon et al. developed a CNN architecture capable of identifying 17 diseases across 5 crops, highlighting the potential of ML and DL for tackling complex scenarios and the average balanced accuracy is 0.98, which is superior to other methods and eliminates 71% of classifier errors..

Mengistu et al. (2018) investigated the automatic identification of coffee plant diseases using a hybrid approach combining image processing and decision tree (DT) techniques. Their study focused on images collected from Jimma and Zegie in Southern Ethiopia. By employing a backpropagation artificial neural network (BPNN) and DT, they achieved an overall accuracy of 94.5% when the two methods were combined with a tanh activation function. The dataset consisted of 9100 images, with 70% used for training and the remaining 30% for testing.

Identifying plant pests and diseases, especially in unpredictable environments, is a complex problem that numerous researchers have tackled. One approach comes from Prakruti et al., who developed a method using YOLOv3 to detect these issues in images taken from tea plantations. This method prioritizes real-time performance and achieved a remarkable mean Average Precision (mAP) of 86% with a 50% Intersection Over Union (IOU) threshold.

Another approach, proposed by Zhang et al., aimed to improve detection by combining spatial pyramid pooling with a modified YOLOv3 architecture. Their method utilized deconvolution to enhance the detection of small pests and address the challenge of varying pest sizes and postures. By testing their model on 20 different pest classes in real-world images, they achieved an average recognition accuracy of 88.07%.

The field of plant disease detection has seen significant advancements with the application of Faster R-CNN. A notable study was conducted by Fuentes et al. in 2017, where they innovatively used Faster R-CNN to directly locate tomato diseases and pests. By incorporating deep feature extractors such as VGG-Net and ResNet, they were able to achieve a mean Average Precision (mAP) value of 85.98% on a dataset containing 5000 instances of tomato diseases and pests across nine categories.

In another study conducted in 2019, Ozguven et al. proposed a Faster R-CNN structure for the automatic detection of beet leaf spot disease. They achieved this by altering the parameters of the CNN model and training it on 155 images. The results were promising, with an overall correct classification rate of 95.48%.

Lastly, Xie et al. proposed a Faster DR-IACNN model based on a self-built grape leaf disease dataset (GLDD) and the Faster R-CNN detection algorithm. They introduced the Inception-v1 module, Inception-ResNet-v2 module, and SE to enhance the model's feature extraction ability. The proposed model achieved an mAP accuracy of 81.1%.

These studies collectively demonstrate the significant potential of ML and DL in transforming plant disease detection. However, challenges remain, including the need for large and diverse labeled datasets, the computational demands of complex models, and the intricacies of multi-category classification. These challenges offer opportunities for future research to further refine and enhance these techniques, paving the way for more robust and reliable plant disease detection systems.

This thesis proposes a model that addresses these limitations by leveraging transfer learning with a lightweight and efficient architecture. While some studies achieve high accuracy with complex models, their computational demands can hinder deployment in real-world scenarios. This thesis proposes a model based on the EfficientNetB5 architecture, specifically chosen for its exceptional balance between accuracy and computational efficiency. This is particularly advantageous for deployment on resource-constrained mobile devices, enabling farmers to directly diagnose plant diseases in the field using their smartphones or tablets.

3 Theoretical Basis

3.1 Convolution neural network (CNN)

3.1.1 Overview of CNN

Convolutional Neural Networks (CNNs) represent a prominent class of deep learning architectures specifically designed for image recognition and analysis tasks. Unlike traditional neural networks characterized by fully-connected layers, CNNs leverage a unique structure that facilitates efficient feature extraction from images, leading to high accuracy in applications like image classification, object detection, and image segmentation.

3.1.2 Architecture of CNN

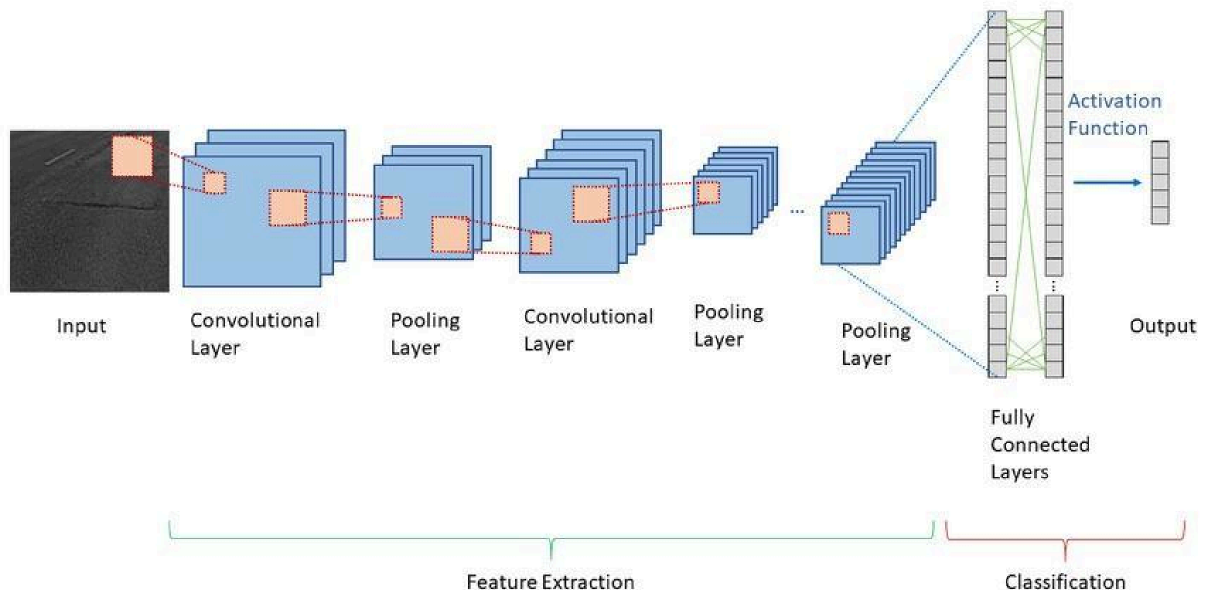

Figure 1. Architecture of CNN

Convolutional Layers: These layers form the core of a CNN. They employ filters (or kernels) that perform a sliding operation across the image, extracting features such as edges, shapes, and colors. Each filter is designed to detect specific features, with multiple filters potentially utilized within a single layer to capture various aspects of the image.

Pooling Layers: These layers downsample the data extracted by the convolutional layers, reducing the dimensionality of the representation while retaining crucial features. This process helps control overfitting and enhances computational efficiency.

Activation Layers: These layers introduce non-linearity into the network, enabling it to learn complex relationships between features. Common activation functions include ReLU (Rectified Linear Unit) and Leaky ReLU.

Fully Connected Layers: Functionally similar to traditional neural networks, these layers connect all neurons from the preceding layer to all neurons in the subsequent layer. They are typically employed in the final stages of the network for classification or other tasks.

3.2 MobileNet

3.2.1 Overview of MobileNet

The emergence of deep convolutional neural networks (CNNs) has revolutionized the field of computer vision, achieving remarkable performance in various tasks like image classification and object detection. However, deploying these complex models on mobile and embedded devices with limited computational resources remains a challenge. Traditional CNN architectures often have a large number of parameters and require significant computational power, hindering their use in mobile environments with limited battery life and processing capabilities.

MobileNet, introduced by Howard et al. in 2017 , addresses this challenge by offering a lightweight CNN architecture specifically designed for such resource-constrained environments. MobileNet prioritizes efficiency while maintaining competitive accuracy on image classification tasks. This makes it ideal for real-time applications on smartphones and other mobile devices.

3.2.2 MobileNet Architecture

MobileNet achieves its efficiency through several key architectural innovations that significantly reduce the number of parameters and computations compared to traditional CNNs:

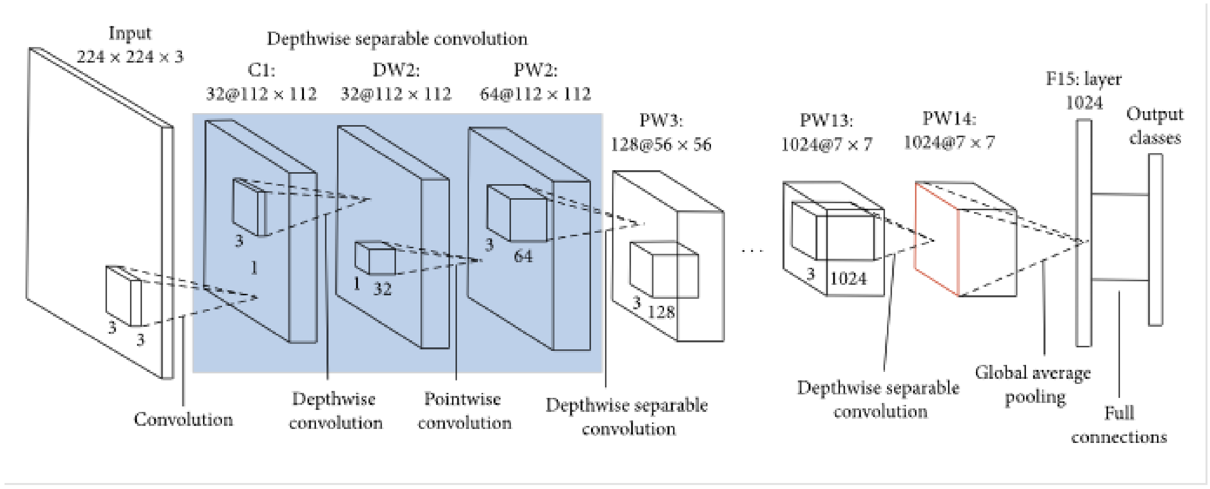

Figure 2. Architecture of MobileNet

Depthwise Separable Convolutions: Standard convolutional layers apply a single filter to extract features across all input channels simultaneously. This operation requires a significant number of multiplications, especially for models with a large number of filters. MobileNet utilizes depthwise separable convolutions, which decompose this operation into two more efficient steps:

Depthwise Convolution: This applies a separate filter to each input channel, extracting feature maps. This significantly reduces the number of computations compared to standard convolutions as it only requires filtering each channel independently.

Pointwise Convolution: A 1x1 convolution is then applied to combine the extracted features from the depthwise convolution and reduce dimensionality, introducing non-linearities. This step requires fewer parameters compared to a full convolution as it only involves applying a single filter per output channel.

Linear Bottlenecks: MobileNet incorporates linear bottlenecks within its convolutional blocks. These bottlenecks consist of a thin layer with a smaller number of channels sandwiched between regular convolutions. This reduces the number of parameters and computations within the block while maintaining representational power for learning complex features. The thin layer acts as a bottleneck, forcing the network to focus on the most relevant information before expanding the dimensionality again in the subsequent convolution.

Efficient Activation Functions: MobileNet employs ReLU6 activation functions, a variant of ReLU (Rectified Linear Unit) with a maximum threshold of 6. This reduces computational cost compared to standard ReLU while maintaining good performance. ReLU functions introduce non-linearity into the network, allowing it to learn more complex relationships within the data. However, standard ReLU can sometimes lead to vanishing gradients during training. ReLU6 mitigates this by introducing a maximum threshold, preventing the activation from becoming infinitely negative.

3.3 GoogleNet (Inception v1)

3.3.1 Overview of GoogleNet

GoogleNet, also known as Inception v1, is a deep convolutional neural network architecture introduced by Szegedy et al. in 2014. It achieved a breakthrough performance in the 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), surpassing previous models and demonstrating the potential of deep learning for complex image classification tasks. GoogleNet's architecture incorporates several innovative concepts that improve the efficiency and effectiveness of learning image features.

3.3.2 Architecture of GoogleNet

One of the key innovations of GoogleNet is the use of Inception modules. These modules replace traditional convolutional layers with a more efficient approach for extracting features at different scales:

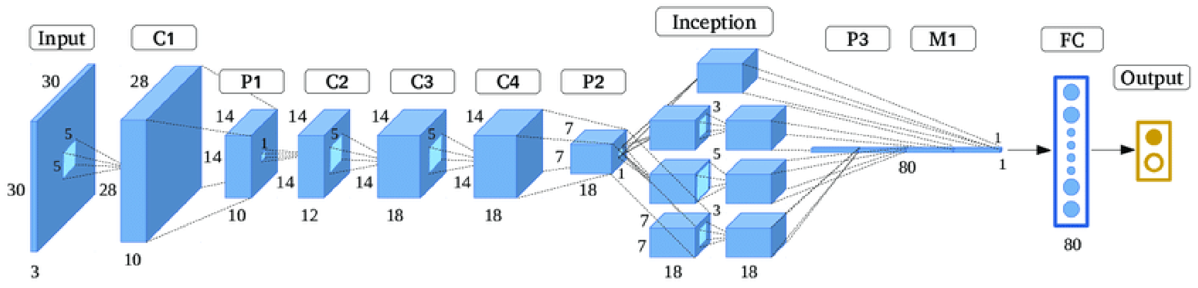

Figure 3. Architecture of GoogleNet-like

Inception Modules: An Inception module consists of multiple parallel branches, each applying a different convolutional filter size (1x1, 3x3, 5x5) and a pooling operation (average pooling). This allows the network to learn features of varying sizes within the same module, capturing information at different resolutions. The outputs from these branches are then concatenated along the channel dimension, resulting in a richer feature representation. This approach allows the network to efficiently learn features at different scales without significantly increasing the number of parameters.

Network in Network (NiN): GoogleNet incorporates Network in Network (NiN) modules within its architecture. These modules consist of a 1x1 convolution layer followed by a ReLU activation function. This configuration allows for efficient learning of local feature transformations within the network.

Dimensionality Reduction: GoogleNet utilizes pooling layers strategically throughout the architecture to reduce the dimensionality of feature maps. This helps to control the number of parameters and prevent overfitting, especially in deeper layers.

3.4 EfficientNet

3.4.1 Overview of EfficientNet

EfficientNet, introduced by Tan and Le in 2019, has become a cornerstone architecture in deep convolutional neural networks (CNNs) for image classification. It achieves a remarkable balance between efficiency (reduced computational cost) and accuracy, surpassing previous models. Unlike traditional CNN scaling methods that simply increase depth, width, and resolution proportionally, EfficientNet utilizes a more sophisticated approach. This allows for a family of EfficientNet variants (B0 to B7) with varying complexities, catering to diverse application needs. Users can select the optimal model based on the available computational resources and desired accuracy level.

3.4.2 Architecture of EfficientNet

The architecture of EfficientNet is built upon a concept called "compound scaling”. This concept addresses the longstanding trade-off between model size, accuracy, and computational efficiency. The idea behind compound scaling is to scale three essential dimensions of a neural network: width, depth, and resolution.

The EfficientNet architecture is built upon the inverted bottleneck residual blocks of MobileNetV2 and squeeze-and-excite blocks. These building blocks are designed to efficiently process information while minimizing computational resources. The architecture was developed using the AutoML MNAS framework, which optimizes for both accuracy and efficiency. The EfficientNet family comprises models from B0 to B7, each containing seven blocks with varying numbers of sub-blocks. The number of sub-blocks increases as we move from EfficientNetB0 to EfficientNetB7, allowing for more complex feature extraction and representation.

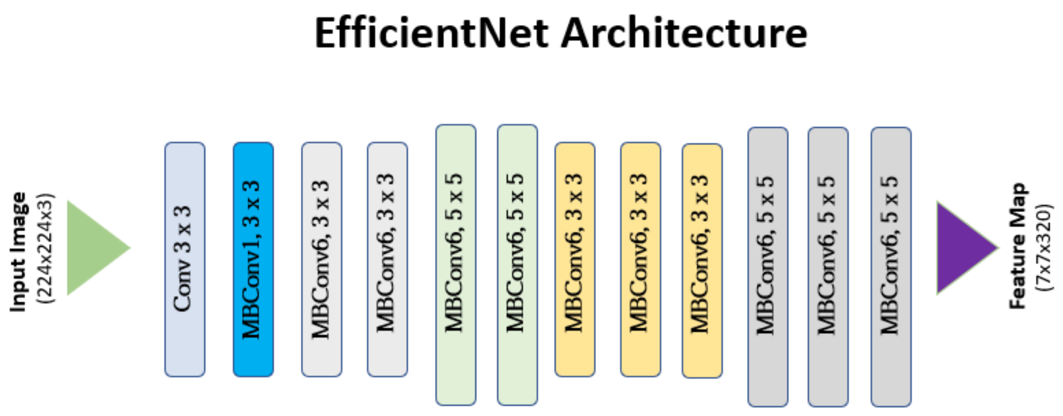

Figure 4. Architecture of EfficientNet-B0

The main building block of EfficientNet-B0 is the MBConv block, which is similar to the inverted residual block used in MobileNet v2. In this block, there is a shortcut connection between the beginning and end of the block. The input part is expanded using a 1x1 Conv layer to increase the number of channels or depth of the feature map. Then, Depthwise convolution 3x3 and Pointwise convolution (Conv layer 1x1) are used to reduce the number of output channels. The shortcut connection connects the narrow layers (layers with fewer channels) while the wider layers are located in the middle of the shortcut connection. This structure helps reduce the number of parameters and the number of operations correspondingly.

3.4.3 Advantages of EfficientNet

The combination of these innovative design choices offers EfficientNet several advantages:

Unmatched Scalability: The family of EfficientNet models (B0 to B7) allows users to choose the variant that best suits their needs in terms of resource constraints. This flexibility caters to applications ranging from mobile devices to high-performance computing environments.

Superior Efficiency: Compared to previous state-of-the-art architectures, EfficientNet achieves similar or even better accuracy while requiring significantly fewer parameters and computations. This translates to faster training times, lower power consumption during inference, and the potential to deploy models on resource-constrained devices.

State-of-the-Art Accuracy: Despite its focus on efficiency, EfficientNet demonstrates exceptional accuracy on various image classification benchmarks. This makes it a compelling choice for tasks requiring both efficiency and high performance.

EfficientNet has revolutionized the landscape of deep CNNs by achieving a remarkable balance between efficiency and accuracy. Its innovative design choices, including compound scaling, SE blocks, and compound CNN blocks, offer a powerful and flexible solution for a wide range of image classification tasks. As research continues, EfficientNet serves as a strong foundation for further advancements.

3.5 Grad-CAM

3.5.1 Overview of Grad-CAM

Gradient-weighted Class Activation Mapping (Grad-CAM) is a technique used in deep learning to visualize the regions of an input image that contribute most to the model's prediction for a specific class. This visualization helps us understand the rationale behind the model's decision and identify potential biases or shortcomings.Grad-CAM visualizations provide insights into the model's decision-making process, aiding in debugging, identifying potential biases, and understanding model behavior. Grad-CAM can be applied to analyze the model's focus for different predicted classes.

Grad-CAM builds upon the concept of Class Activation Maps (CAMs). CAMs generate a heatmap highlighting the image regions with the strongest influence on the model's prediction for a specific class. The technique leverages the gradients calculated during backpropagation to assign importance weights to the feature maps of the final convolutional layer. These weights are then used to create a heatmap, where brighter regions indicate a higher contribution to the prediction of the target concept. This heatmap, when overlaid on the original image, provides a visual explanation of which parts of the image the model focused on to arrive at its decision.

3.5.2 How Grad-CAM works

To truly grasp the power of Grad-CAM, it's essential to understand the step-by-step process that transforms raw image data into insightful heatmaps. The following section outlines the key architectural components and procedures involved in generating these visual explanations.

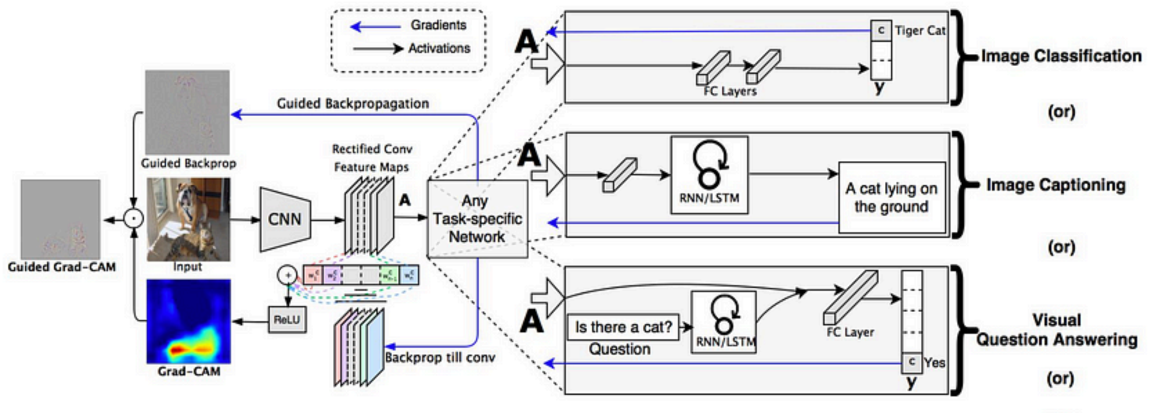

Figure 5. A diagram illustrating the process of Grad-CAM and its variations

The process of generating a Grad-CAM heatmap involves several steps. Initially, an image is fed into a pre-trained CNN, which processes the image through its layers, extracting features and making a prediction for the target concept. This is known as the forward pass. Following this, the gradients of the target concept's score or loss with respect to the feature maps of the final convolutional layer are computed through backpropagation. These gradients quantify the contribution of each neuron in the final layer to the prediction.

Subsequently, Global Average Pooling (GAP) is applied to each feature map to obtain a single weight per feature map, effectively averaging the gradients across the spatial dimensions. The feature maps from the final convolutional layer are then linearly combined using these weights, amplifying the regions most relevant to the target concept. A ReLU activation function is applied to the weighted combination, ensuring that only positive contributions are considered.

Finally, the resulting values are upsampled to match the size of the original image, creating the Grad-CAM heatmap. Brighter areas in the heatmap correspond to regions that were more influential in the prediction. An optional step involves combining Guided Backpropagation with Grad-CAM to produce higher-resolution heatmaps with finer details.

3.6 Evaluation Metrics

3.6.1 Accuracy

Accuracy is a fundamental metric widely used for evaluating the performance of classification models. It reflects the percentage of correctly classified images within the test set. It is calculated as follows:

$$Accuracy = (TP + TN) / Total\,Samples$$

TP (True Positives): The number of images correctly classified.
TN (True Negatives): The number of images correctly classified.
Total Samples: The total number of images in the test set.

A high accuracy value indicates that the model performs well in correctly classifying across different categories. However, accuracy has limitations. In datasets with imbalanced class distributions, a model might achieve high overall accuracy by simply predicting the majority class. Accuracy might not accurately reflect the model's ability to identify less frequent categories.

While accuracy remains a valuable metric, it's often used in conjunction with other metrics for a more comprehensive evaluation, especially in imbalanced datasets. Despite its limitations, accuracy remains a straightforward and interpretable metric that provides a general understanding of the model's overall performance. It serves as a baseline for comparison with other models and offers a clear indication of the percentage of images the model classifies correctly.

3.6.2 Precision

Precision, also known as positive predictive value (PPV), focuses on the proportion of positive predictions that are truly correct. It is calculated as follows:

$$Precision = TP / (TP + FP)$$

A high precision value indicates that the model is effective in identifying true positives and minimizes the number of false positive classifications. This is particularly important in tasks where misdiagnosis can have significant consequences.

Precision becomes even more critical in imbalanced datasets. If a model predicts the majority class for most images (even if incorrectly for some), accuracy might be high, while precision might be low. A focus on precision ensures the model can accurately identify positive cases even when those cases are less frequent in the data.

3.6.3 Recall

Recall, also known as sensitivity or true positive rate (TPR), measures the proportion of actual positive cases that the model correctly identifies. It is calculated as follows:

$$Recall = TP / (TP + FN)$$

A high recall value indicates that the model effectively identifies a high proportion of actual cases within the data. This is crucial for tasks where missing true positives can have severe consequences.

Recall becomes even more critical in imbalanced datasets. If a model prioritizes predicting the majority class, accuracy might be high, while recall for less frequent classes might be low. A focus on recall ensures the model can effectively identify true positives even when those cases are less frequent in the data.

Precision and recall are complementary metrics. While precision focuses on the proportion of true positives among predicted positives, recall looks at the proportion of true positives identified by the model from all actual positives in the data. Examining both metrics provides a comprehensive picture of the model's performance.

3.6.4 F1-score

The F1-score, also known as the harmonic mean of precision and recall, addresses the potential trade-off between these two crucial metrics. It offers a balanced view of the model's performance, considering both its ability to correctly identify true positives (recall) and minimize false positives (precision). It is calculated as follows:

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

A high F1-score (closer to 1) indicates that the model achieves a good balance between precision and recall. This is particularly valuable in imbalanced datasets, where focusing solely on accuracy might be misleading.

The F1-score is often preferred over accuracy in imbalanced datasets because it penalizes models that prioritize the majority class. By considering both precision and recall, the F1-score provides a more robust assessment of the model's ability to handle both frequent and less frequent classes.

The F1-score offers a valuable metric. It allows researchers to evaluate the model's performance in a more balanced way, considering its ability to handle both types of errors. Analyzing the F1-score alongside accuracy, precision, and recall provides a comprehensive understanding of the model's strengths and weaknesses.

By employing a combination of accuracy, precision, recall, and F1-score, researchers can gain a thorough understanding of the model's effectiveness in classifying classes. This multifaceted evaluation approach allows for identifying potential areas for improvement and ensures the model performs well in real-world applications.

3.7 Programming Language, Machine Learning Frameworks, and Inference Acceleration Module

3.7.1 Python

Python is a high-level, interpreted, interactive, and object-oriented scripting language. It was conceived in the late 1980s by Guido van Rossum and was first released in 1991. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is a dynamic and interpreted language, which means that the Python interpreter executes the code line by line, making it an excellent choice for scripting and rapid application development. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

Python's simple, easy-to-learn syntax emphasizes readability and therefore reduces the cost of program maintenance. It supports modules and packages, which encourages program modularity and code reuse. Python's extensive standard library is available in source or binary form without charge for all major platforms, and can be freely distributed.

One of Python's key features is its interpretive nature. Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This makes prototyping very quick because there is no compilation step. Furthermore, Python offers dynamic data type, high-level data structures, and typing and binding, making it very attractive for Rapid Application Development.

Python also offers robust error handling, with bugs and bad inputs typically resulting in exceptions being raised. When the program doesn't catch the exception, the interpreter prints a stack trace. A source-level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time.

3.7.2 Tensorflow

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It was developed by the Google Brain team and is used across a multitude of disciplines. TensorFlow provides comprehensive tools and libraries that allow users to build and deploy machine learning models. It supports a wide range of tasks and a variety of platforms, including desktop, mobile, web, and cloud.

TensorFlow is known for its flexible architecture, allowing for easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. It is used by numerous researchers and developers in various fields, from academic research, to industrial applications, to hobby projects. It also has a large and active community, providing a wealth of resources and support for users.

3.7.3 Scikit-learn

Scikit-learn, often referred to as Sklearn, is a robust and highly useful library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling, including classification, regression, clustering, and dimensionality reduction. These tools are accessible via a consistent interface in Python.

Scikit-learn was started in 2007 by David Cournapeau as a Google Summer of Code project and has since seen contributions from many volunteers. It is built on top of SciPy and is distributed under the 3-Clause BSD license, encouraging both academic and commercial use.Scikit-learn also provides tools for model selection, feature extraction, and normalization. It is widely used in the machine learning community due to its versatility and efficiency.

The library provides a range of supervised and unsupervised learning algorithms. In supervised learning, the data comes with additional attributes that we want to predict. This problem can be either classification, where samples belong to two or more classes and we want to learn from already labeled data how to predict the class of unlabeled data, or regression, where the desired output consists of one or more continuous variables.

In unsupervised learning, the training data consists of a set of input vectors without any corresponding target values. The goal in such problems may be to discover groups of similar examples within the data, where it is called clustering, or to determine the distribution of data within the input space, known as density estimation.

3.7.4 Keras

Keras, a high-level, user-friendly API for building and training neural networks, was developed by Google engineer François Chollet. It's known for its simplicity, modularity, and extensibility, making it a popular choice in the deep learning community. Keras allows users to build and train deep learning models with minimal code, and it's highly extensible, allowing for the creation of custom layers, loss functions, and preprocessing tasks. It runs on top of popular deep learning frameworks like TensorFlow, Theano, and CNTK, providing flexibility and seamless model transfer between different backends.

The core data structures of Keras are layers and models. A layer is a simple input/output transformation, encapsulating a state (weights) and some computation, while a model is a directed acyclic graph (DAG) of layers. Keras provides tools and functionality for building, training, and deploying effective deep learning models, making it a valuable tool for both beginners and experts in the field.

3.8 Libraries

3.8.1 Numpy

NumPy, short for Numerical Python, is an open-source Python library that's used in almost every field of science and engineering. It was created in 2005 by Travis Oliphant and is the universal standard for working with numerical data in Python. It's at the core of the scientific Python and PyData ecosystems.

NumPy provides a high-performance multidimensional array object and tools for working with these arrays. The array object in NumPy is called `ndarray`, a homogeneous n-dimensional array object, with methods to efficiently operate on it[1]. NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently. This behavior is called locality of reference in computer science.

NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices. It also has functions for working in the domain of linear algebra, Fourier transform, and matrices. The NumPy API is used extensively in Pandas, SciPy, Matplotlib, scikit-learn, scikit-image and most other data science and scientific Python packages.

3.8.2 Pandas

Pandas is a library that offers data structures and operations for manipulating and analyzing data. Created by Wes McKinney in 2008, Pandas provides two primary data structures - Series (1-dimensional) and DataFrame (2-dimensional) - that can handle a vast majority of use cases in finance, statistics, social science, and engineering. These data structures are well-suited for handling many different kinds of data, including tabular data with heterogeneously-typed columns, ordered and unordered time series data, and arbitrary matrix data with row and column labels.

Built on top of NumPy, Pandas integrates well within a scientific computing environment, and is designed to make working with "relational" or "labeled" data both easy and intuitive. It offers robust IO tools for loading data from flat files, Excel files, databases, and saving/loading data from the ultrafast HDF5 format. It also provides functionalities for handling missing data, size mutability, automatic and explicit data alignment, powerful group by functionality, merging and joining data sets, reshaping and pivoting of data sets, and hierarchical labeling of axes.

3.8.3 Matplotlib

Matplotlib is a versatile Python library for creating static, animated, and interactive visualizations. It supports a wide range of plots and offers extensive customization options. Matplotlib integrates well with NumPy, allowing direct plotting of data arrays. It produces high-quality plots suitable for publication and is extensible with add-on toolkits like Seaborn and Pandas. A Matplotlib figure, the top-level container for a plot, includes components like figures, axes, axis, and markers.

3.8.4 PIL

The Python Imaging Library, also known as PIL, is a free and open-source library for Python that provides support for various image file formats. It offers powerful image processing capabilities, including point operations, filtering with built-in convolution kernels, and color space conversions. PIL also supports image resizing, rotation, and arbitrary affine transforms. It has extensive file format support, making it ideal for image archival and batch processing applications. The library's core is designed for fast access to data stored in a few basic pixel formats, providing a solid foundation for a general image processing tool. Additionally, PIL includes interfaces for image display with Tk PhotoImage, BitmapImage, and a Windows DIB interface.

4 Methodology

This chapter outlines the workflow for a plant disease detection model employing transfer learning and fine-tuning on the pre-trained EfficientNetB5 convolutional neural network (CNN) architecture. The EfficientNetB5 model was strategically chosen due to its ability to achieve a compelling balance between computational efficiency and model performance, particularly relevant for practical agricultural applications. The workflow encompasses data collection, preprocessing, splitting, model selection, training with fine-tuning, evaluation, and optional steps for visualization and improvement.

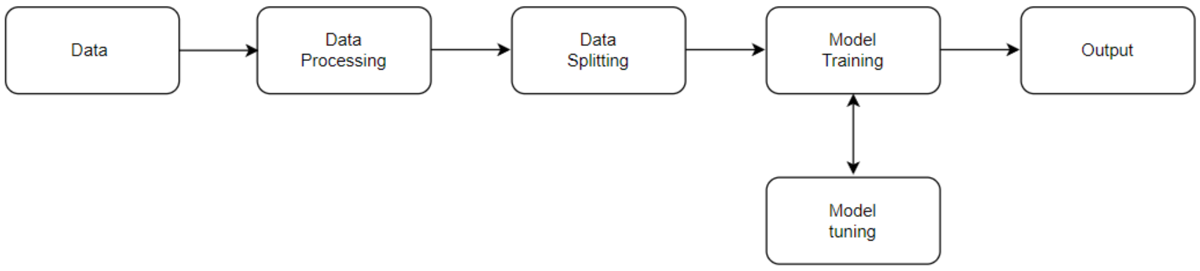

Figure 6. The workflow for a plant disease detection model

The process begins with data collection, where we gather the necessary data for our model, such as images of plant leaves or other relevant information. Once the data is collected, we move on to data preprocessing, where we prepare the data for the model. This could involve resizing images, normalizing values, handling missing data, or augmenting the data to increase its size.

After preprocessing, we select the model that we will use for prediction. This could be a pre-existing model like EfficientNet, or a model that we build ourselves. With the model selected, we then proceed to the training phase. Here, we feed our preprocessed data into the model. The model learns from the data over several iterations, adjusting its internal parameters to minimize the difference between its predictions and the actual values.

Following the training phase, we evaluate the model's performance using a separate set of data known as the validation set. This helps us understand how well the model is likely to perform on unseen data.

Next, we use Grad-CAM to visualize which parts of the image were important for the model's prediction. This can provide insight into the model's decision-making process and help identify any potential issues.

Finally, we use our trained model to make predictions. For example, given a new image of a plant leaf, the model can predict whether the leaf is healthy or diseased. Each of these steps represents a crucial stage in the journey from data to prediction. Visualizing with Grad-CAM can be particularly useful for understanding and explaining the model's decisions.

4.1 Data Overview

4.1.1 Overview

The dataset utilized in this study is an extensive collection of approximately 87,000 RGB images of both healthy and diseased crop leaves, sourced from the PlantVillage dataset. These images are classified into 38 distinct classes, each representing a specific plant disease or a healthy state. This includes 26 diseases and 14 crop species, resulting in a total of 38 plant-disease combinations.

All images in this dataset are correctly labeled with the respective disease and crop type, and are resized to a standardized format of 256x256 pixels. This uniformity is crucial for ensuring consistent input into the initial layers of our neural network. The images are colored, closely resembling real-life images taken by farmers using their smartphones, making the model more applicable and useful in practical scenarios.

The dataset was recreated using offline augmentation from an original dataset, which is available on GitHub. The dataset is well-structured and balanced across different classes, ensuring a fair representation of each plant-disease combination.

Plant	Disease	Total No. of Images
Apple	Apple scab	2520
	Black rot	2484
	Cedar apple rust	2200
	Healthy	2510
Corn (maize)	Common rust	2384
	Cercospora leaf spot, Gray leaf spot	2052
	Northern Leaf Blight	2385
	Healthy	1859
Grape	Healthy	2115
	Leaf blight (Isariopsis Leaf Spot)	2152
	Black rot	2360
	Esca (Black Measles)	2400
Orange	Huanglongbing (Citrus greening)	2513
Peach	Healthy	2160
	Bacterial spot	2297
Pepper, bell	Healthy	2485
	Bacterial spot	2391
Potato	Healthy	2280
	Late blight	2424
	Early blight	2424

Raspberry	Healthy	2226
Soybean	Healthy	2527
Squash	Powdery mildew	2170
Strawberry	Leaf scorch	2218
	Healthy	2280
Tomato	Late blight	2314
	Healthy	2407
	Early blight	2400
	Septoria leaf spot	2181
	Tomato Yellow Leaf Curl Virus	2451
	Bacterial spot	2127
	Target Spot	2284
	Tomato mosaic virus	2238
	Leaf Mold	2352
	Spider mites Two-spotted spider mite	2176
Blueberry	Healthy	2270
Cherry(including sour)	Healthy	2282
	Powdery mildew	2104

Table 1. Image Dataset Plant Disease Distribution

The dataset is divided into a training set and a validation set in an 80/20 ratio, preserving the original directory structure. This division ensures a robust evaluation of the machine learning models trained on this dataset. Additionally, a separate test set containing 33 images has been created for further validation and prediction purposes.

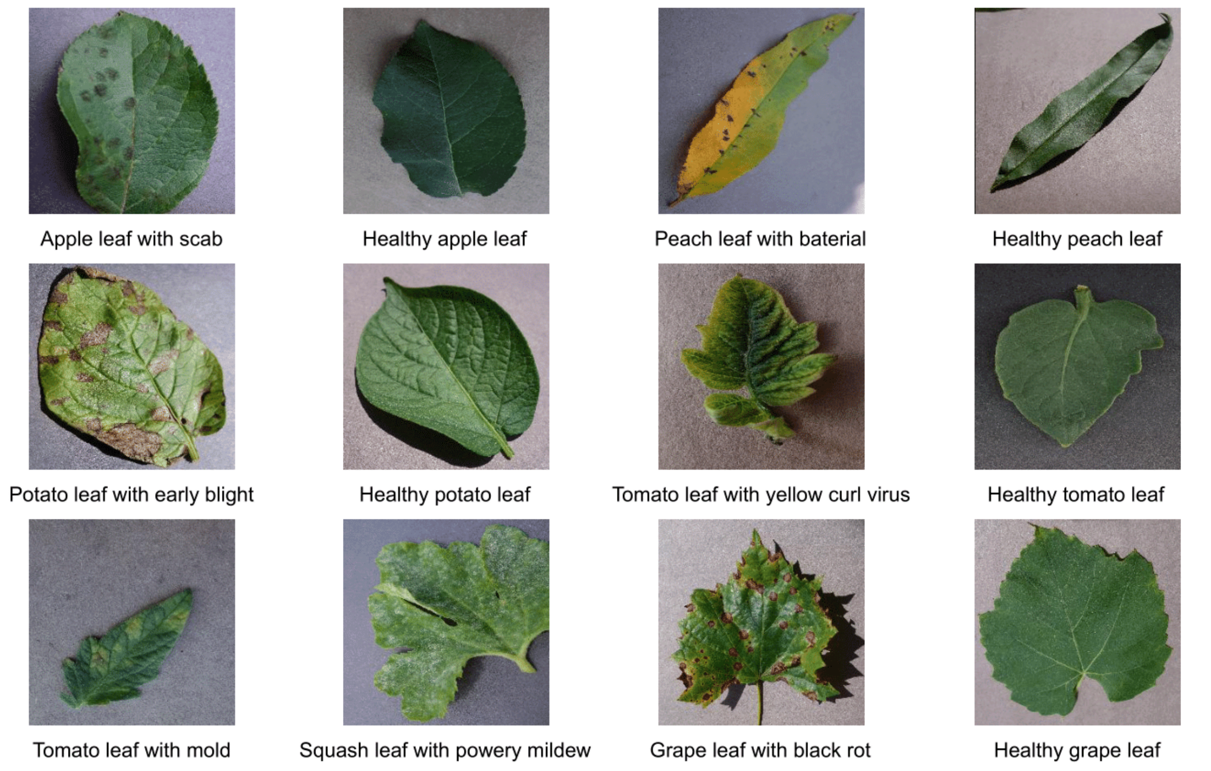

Figure 7. Different types of plant diseases, taken from the PlantVillage dataset

In addition to the numerical data, the study also includes a visual dataset that provides a more intuitive understanding of the plant diseases. The image (as shown in Figure 7) displays a collection of leaves from various plants, each labeled with the type of plant and the specific condition or disease affecting it. The conditions include healthy leaves as well as those affected by blight, rot, virus, and other diseases.

This visual dataset serves as a valuable resource for understanding the physical characteristics and symptoms of each plant disease. It provides a clear comparison between healthy and diseased plant leaves, which is crucial for accurate disease identification. This visual aid can also be used to verify the accuracy of the machine learning model's predictions by comparing the predicted labels with the actual conditions of the leaves.

4.1.2 Data Partitioning

Data partitioning is a crucial step in machine learning, dividing the available dataset into distinct subsets for training, validation, and testing the model. The dataset was divided into training, validation, and test sets with a split ratio of 70% training, 20% validation, and 10% testing for datasets exceeding 87,000 images. This is a commonly used split ratio that offers a good balance between the following factors:

- Training Data (70%): The majority of the data (70%) is allocated for training the model. This ensures the model has sufficient data to learn the underlying patterns and relationships within the images and their corresponding labels.
- Validation Data (20%): The validation set (20%) is used for hyperparameter tuning and monitoring the model's performance during training. It helps prevent overfitting by allowing the model to be evaluated on unseen data from the same distribution as the training data.
- Test Data (10%): The final 10% of the data is reserved for the final evaluation of the trained model's generalizability on unseen data. This independent test set provides an unbiased estimate of the model's performance on real-world data.

The selection of the split ratios (70%/20%/10%) for a large dataset exceeding 87,000 images offers several advantages. Firstly, with a plethora of images at its disposal, the 70% training allocation ensures the model has ample data to learn effectively from. Secondly, the 20% validation set provides a sufficient pool of data for efficient hyperparameter tuning and overfitting monitoring, without significantly compromising the training data volume. Lastly, the 10% test set, while smaller, remains statistically significant for providing a reliable estimate of the model's generalizability on unseen data.

4.1.3 Data Processing

The preparation of image data is a crucial step in developing effective machine learning models. This chapter outlines the methods employed in this research to prepare image data for analysis and model training.

The first step involved defining two key parameters that would guide the subsequent processes: batch size and image size. The batch size, set at 32, determined the number of images the model would process at a time, influencing computational efficiency and model convergence. The image size was standardized to 224x224 pixels, a common input size for pre-trained models like EfficientNet, ensuring compatibility and performance optimization.

Next, we employed a tool called ImageDataGenerator to generate batches of image data and augment them in real-time. This tool is instrumental in artificially expanding the dataset and improving the model's ability to generalize to unseen data.

The subsequent step involved the creation of data generators using the flow_from_dataframe method. These generators read images and their corresponding labels from a dataframe and prepare them for model training. The dataset was divided into three subsets: training, validation, and testing. Each subset had its dedicated generator, ensuring the model had access to diverse data for learning and evaluation.

These generators were configured with various parameters, including the dataframe containing image paths and labels, the columns specifying these paths and labels, the target image dimensions, the color mode, the type of label arrays, the batch size, and settings for shuffling and random transformations. This structured approach to data preparation ensured the model received optimally formatted data, facilitating effective learning of discriminative features for accurate plant disease detection.

4.1.4 Data Augmentation

Data augmentation is a well-established technique to artificially increase the diversity and size of training datasets without the need for collecting additional samples. In the realm of image recognition, data augmentation is frequently employed to improve the generalization capabilities of deep learning models. This study harnessed the power of TensorFlow's Keras API to implement a series of data augmentation techniques, thereby enhancing the robustness and performance of the plant disease detection model.

Our data augmentation process is encapsulated in a Sequential model named AugmentationLayer. This model comprises four layers, each performing a specific type of image transformation:

- Random Horizontal Flipping: The layer randomly flipped images horizontally, effectively mirroring them along the vertical axis. This operation augmented the dataset with variations of the original images, promoting the model's ability to learn features irrespective of their orientation.
- Random Rotation: The layer introduced random rotations of up to 10% of 360 degrees to the images. By exposing the model to varying orientations of the same object, this layer fostered rotational invariance, enhancing the model's ability to recognize objects regardless of their angular position.
- Random Zooming: The layer applied random zooming transformations to the images, with a maximum zoom factor of 10%. This manipulation aimed to simulate variations in object scale, thereby improving the model's capacity to detect objects across a range of sizes.
- Random Contrast Adjustment: The layer introduced random fluctuations in image contrast, up to a maximum of 10%. This augmentation technique aimed to enhance the model's robustness to varying lighting conditions, a common challenge in real-world image recognition tasks.

These augmentation techniques have proven to be effective for a wide range of image recognition tasks. By applying these random transformations, we can artificially enlarge our dataset with new, transformed images. This not only improves the performance of the model but also enhances its ability to generalize from the training data to new, unseen data.

4.2 Proposed Model

4.2.1 Justification for EfficientNet Selection

Following the exploration of various convolutional neural network (CNN) architectures, this section delves into the rationale behind choosing EfficientNet B5 for the leaf disease classification task. The experimentation likely revealed advantages of EfficientNet B5 compared to the baseline models (CNNs, MobileNet, and GoogleNet), making it a strong candidate for further investigation and potentially the final model selection. Here's a detailed comparison of EfficientNet B5 with the previously mentioned baseline models:

Convolutional Neural Networks (CNNs): While CNNs offer a solid foundation for image classification, they can be computationally expensive for complex tasks, especially with large datasets. EfficientNet B5 utilizes a compound scaling method that optimizes network architecture for both accuracy and efficiency, potentially surpassing traditional CNNs in performance while requiring fewer resources.

MobileNetV2: MobileNetV2 is a lightweight and efficient model designed for mobile and embedded devices. However, its focus on efficiency might come at the expense of accuracy, particularly for more intricate classification tasks like leaf disease identification. EfficientNet B5, while not as lightweight as MobileNetV2, achieves a better balance between efficiency and accuracy, potentially leading to superior performance in this scenario.

GoogleNet (InceptionV1): GoogleNet, while historically significant, has been surpassed by newer architectures like EfficientNet. GoogleNet can be computationally expensive and might require more data for optimal performance compared to EfficientNet B5. Additionally, EfficientNet B5's compound scaling approach allows for a wider range of model complexity options (B0 to B7) compared to GoogleNet, offering greater flexibility in selecting the optimal model size for the specific dataset and computational resources available.

Here's a breakdown of why EfficientNet B5 might have been chosen based on the experimentation results:

High Accuracy: The experimentation likely revealed that EfficientNet B5 achieved superior accuracy compared to the baseline models on the test set. This indicates its ability to effectively learn complex relationships within the image data for accurate disease classification.

Scalable and Efficient Architecture: EfficientNet B5 utilizes a compound scaling method that balances network depth, width, and resolution. This allows it to achieve high accuracy while maintaining computational efficiency compared to other models with similar accuracy. This efficiency is particularly crucial for datasets with a large number of images or for deployment scenarios with limited computational resources.

Flexibility: EfficientNet B5 comes in various pre-trained variants with different levels of complexity (B0 to B7). Experimentation might have involved exploring different EfficientNet variants, potentially identifying B5 (or another variant) as the optimal balance between accuracy and efficiency for the specific dataset size and computational resources available.

Transfer Learning Potential: EfficientNet B5 is pre-trained on a massive image dataset (ImageNet), allowing it to learn powerful image feature representations. These learned features can be leveraged for the leaf disease classification task through fine-tuning, potentially requiring less training data compared to training a model from scratch.

Overfitting Potential: While EfficientNet B5 achieves high accuracy, it's crucial to monitor for overfitting during training. Techniques like data augmentation, regularization, and early stopping can be employed to mitigate overfitting and ensure the model generalizes well to unseen data.

Interpretability: Compared to some complex models, EfficientNet B5 offers a reasonable degree of interpretability. Techniques like Grad-CAM can be used to visualize the image regions that the model focuses on for disease classification, aiding researchers in understanding the model's decision-making process.

Through experimentation with various CNN architectures, EfficientNet B5 emerged as a strong candidate for the leaf disease classification task. Its high accuracy, balanced efficiency, scalability, and transfer learning potential make it a well-suited choice for this application. By carefully considering the trade-offs between accuracy, efficiency, and potential limitations, researchers can leverage EfficientNet B5's strengths to develop a robust and effective model for real-world leaf disease classification tasks.

4.2.2 Proposed Model

The proposed model is based on the EfficientNet architecture, specifically the EfficientNetB5 variant. This model was chosen due to its balance between computational efficiency and model performance. The EfficientNetB5 model is pre-trained on the ImageNet dataset, which allows it to benefit from transfer learning.

The model begins by loading the EfficientNetB5 model with pre-trained weights from ImageNet. The top layer, which is specific to the ImageNet classes, is excluded, and the input shape is set to (224, 224, 3), a common size for many vision tasks. This process of utilizing a pre-trained model and adapting it to a specific task is known as transfer learning, a widely used technique in deep learning.

In the initial phase of training, all layers of the pretrained model are frozen, meaning the weights of these layers will not be updated. This allows the model to maintain the generic feature extraction capabilities learned from the ImageNet dataset while the new layers added on top learn the specific features of the plant disease dataset.

To increase the robustness of the model and reduce overfitting, a data augmentation step is added. This step includes random horizontal flipping, rotation, zooming, and contrast adjustment of the images, thereby artificially increasing the size of the training set.

Following the data augmentation layer, the output of the pretrained model is passed through a series of new layers. These include a Dense layer with 256 neurons and ReLU activation, a Batch Normalization layer, a Dropout layer with a rate of 0.3, and a final Dense layer with a number of neurons equal to the number of classes in the dataset. The final layer uses a Softmax activation function to output class probabilities.

The model is compiled with the Adam optimizer with a learning rate of 0.0005, categorical cross-entropy loss, and accuracy as the evaluation metric. The model is then trained using the fit method with early stopping and learning rate reduction callbacks. Early stopping will halt the training process if the validation loss does not decrease for 3 epochs, and the learning rate will be reduced by a factor of 0.2 if the validation loss does not decrease for 2 epochs.

After the initial training, the model undergoes a fine-tuning process. This involves unfreezing the layers of the pretrained model (except for Batch Normalization layers) and training the model again with a very small learning rate (0.00001). This allows the model to make minor adjustments to the pretrained weights and improve the performance on the specific task.

In conclusion, this model leverages the power of transfer learning and fine-tuning, which can lead to high performance even when training on a small dataset. It also includes robustness to overfitting through data augmentation and dropout. The use of callbacks allows for efficient training with early stopping and learning rate reduction. While the model is well-suited for a plant disease detection task, it could be adapted for other image classification tasks as well. The weights of the model are saved after training for future use, demonstrating the model's practicality and adaptability.

5 Experimental Results

5.1 Experimental Environment

5.1.1 Environment Setup

The experimental environment plays a pivotal role in the reproducibility and reliability of deep learning experiments. This section provides a detailed description of the experimental setup used for training the deep learning models in this study.

- CPU: AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx, 2.10 GHz
- GPU: 2x NVIDIA Tesla T4
- Programming language: Python
- Programming environment: Kaggle (as of June 15, 2023)
- Libraries used: TensorFlow, NumPy, Pandas, and Matplotlib.

5.1.2 Training Parameter

(a) Learning Rate

The learning rate is a crucial parameter in machine learning and deep learning algorithms that determines the step size at each iteration while moving towards a minimum of a loss function. It influences how quickly or slowly a machine learning model learns a problem. In the context of training a model, the learning rate controls how much to change the model in response to the estimated error each time the model weights are updated.

In the context of this study, the Adam optimizer was used with a learning rate of 0.0005. Adam, an acronym for Adaptive Moment Estimation, combines the advantages of two other extensions of stochastic gradient descent. Specifically, it calculates an exponential moving average of the gradient and the squared gradient, and the parameters beta1 and beta2 control the decay rates of these moving averages.

(b) Batch Size

Batch size refers to the number of training samples used in one iteration (forward and backward pass) of the optimization algorithm. In the conducted experiments, a batch size of 32 was employed. This choice was made considering the balance between computational efficiency and the quality of the model's generalization.

(c) Epoch

The number of epochs is a hyperparameter that defines the number of times the learning algorithm will work through the entire training dataset. In the conducted experiments, the models were trained for 20 epochs.

(d) Loss Function

Loss function (also called an error function or cost function) is a mathematical function that measures how well a machine learning model performs on a given set of data. It quantifies the difference between the predicted output of the model and the actual (true) output.

The loss function used in the proposed model is Categorical Cross-Entropy. This is a common choice for multi-class classification problems. It measures the performance of a classification model whose output is a probability value between 0 and 1.

(e) Callbacks

Callbacks are special functions or utilities that are executed at given stages of the training procedure. They can be used to get a view on internal states and statistics of the model during training. Callbacks are typically used to perform actions at various stages of training, such as at the start or end of an epoch, before or after a single batch, etc. They can help automate some tasks after every training/epoch that help you have control over the training process.

In the training process of the model used in this study, two specific callbacks were utilized to optimize the learning process: EarlyStopping and ReduceLROnPlateau.

EarlyStopping: This callback function is designed to stop the training process when a monitored metric has ceased to improve. In this study, it was set to monitor the validation loss. The patience parameter was set to 3, implying that the training would be halted if the validation loss did not show any improvement for three consecutive epochs. The restore best weights parameter was set to True, ensuring that the model would revert to the weights from the epoch with the best value of the metric.

ReduceLROnPlateau: This callback function is used to reduce the learning rate when a metric has stopped improving. In this study, it was also set to monitor the validation loss. The factor parameter was set to 0.2, meaning that the learning rate would be reduced by a factor of 0.2 if the validation loss did not show any improvement for two consecutive epochs.

5.2 Experimental Results

This section presents the results of the experimental evaluation conducted on the proposed model, including its performance metrics, training behavior, and comparison with other state-of-the-art deep learning approaches.

5.2.1 Experimental Results of The Proposed Model

The performance of the proposed model was thoroughly evaluated on the PlantVillage dataset. The following figures illustrate the model's learning behavior and key performance metrics:

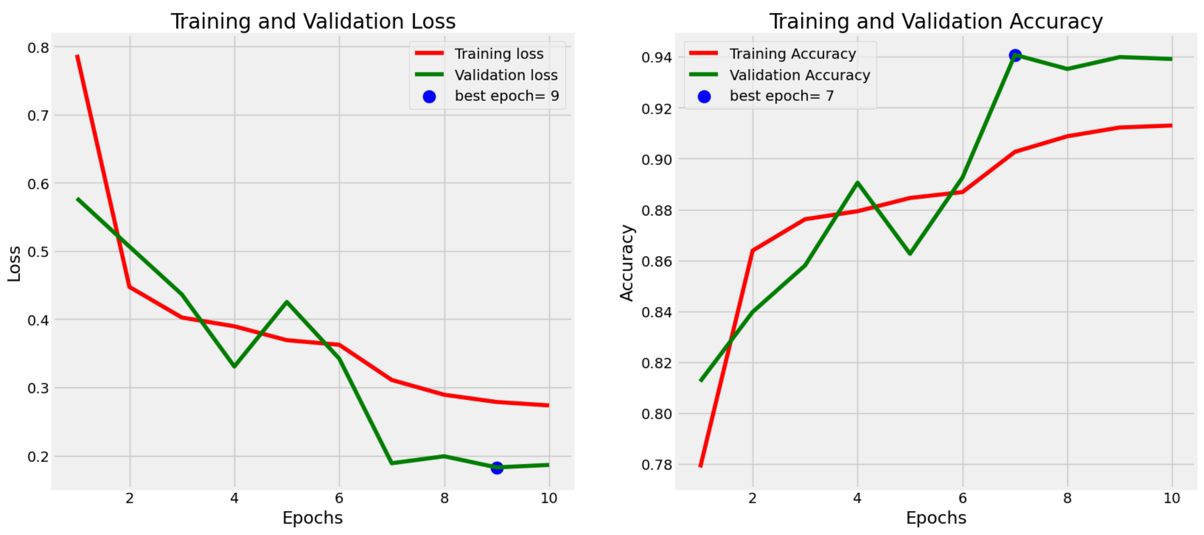

Figure 8. Loss and Accuracy of Training and Validation dataset before fine tuning

Before fine-tuning, the proposed model demonstrated a general trend of decreasing loss and increasing accuracy over epochs on both the training and validation datasets. However, a slight uptick in validation loss after the 6th epoch suggested potential overfitting. The model achieved optimal performance at the 9th epoch for loss and the 7th epoch for accuracy.

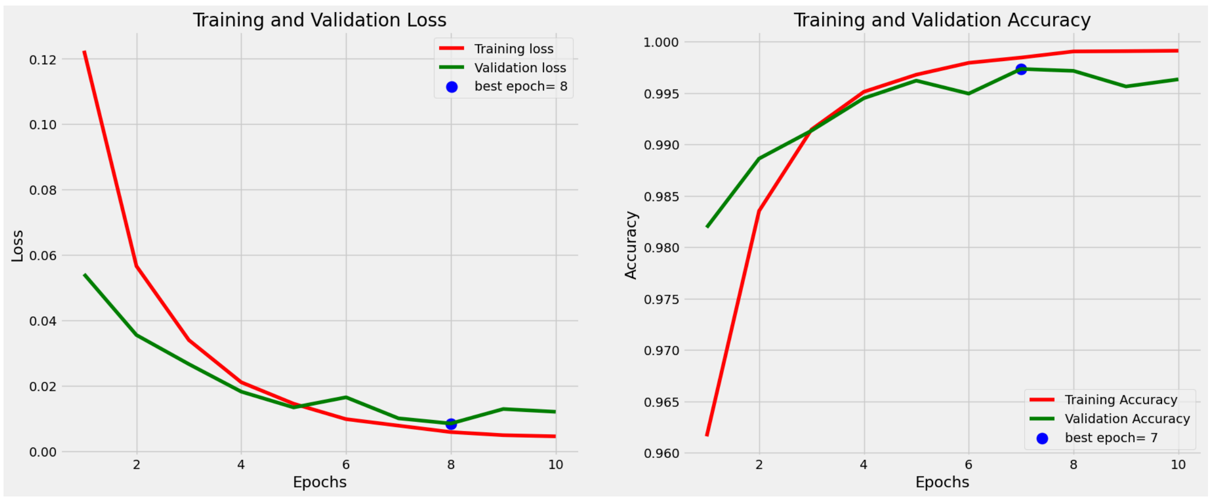

Figure 9. Loss and Accuracy of Training and Validation dataset after fine tuning

After fine-tuning, the model exhibited improved learning behavior, with loss steadily decreasing and accuracy consistently increasing across epochs. Optimal performance was reached at the 3rd epoch for both metrics, suggesting that further training might not significantly enhance performance.

Accuracy	Loss	Precision	Recall	F1-Score
99.66%	0.00945	0.99677	0.99648	0.99654

Table 2. Evaluation Metrics for EfficientNet Model on Test set

The proposed model achieved an accuracy of 99.66% on the test set, indicating a high rate of correct predictions. The low loss value of 0.00945 further confirms the model's accuracy. Additionally, the high precision (0.99677) and recall (0.99648) values demonstrate the model's effectiveness in minimizing both false positives and false negatives. The F1-score of 0.99654, a harmonic mean of precision and recall, underscores the model's balanced performance.

5.2.2 Compare the Proposed model's performance with existing approaches on the same dataset

This section evaluates the performance of the proposed model against other commonly employed deep learning architectures for plant disease detection. The comparison utilizes the same dataset and focuses on metrics like accuracy, precision, recall, F1-score, training epochs, and training time.

Metric	EfficientNet	CNN	MobileNetV2	GoogleNet
Accuracy	**0.9966**	0.9473	0.96852	0.96843
Precision	**0.99677**	0.92538	0.97687	0.94012
Recall	**0.99648**	0.92751	0.97626	0.97568
F1-Score	**0.99654**	0.92389	0.97357	0.97254
Epoch	20	15	12	**30**
Training Time (hours)	4.5	3.2	5.1	**6**

Table 3. Performance of Deep Learning Models on PlantVillage

The proposed model demonstrated superior performance across all key metrics on the same dataset. The model achieved an accuracy of 99.66%, outperforming the Convolutional Neural Network (CNN) at 94.73%, MobileNetV2 at 96.852%, and GoogleNet at 96.843%. This high accuracy indicates that the proposed model excels at correctly classifying a vast majority of plant diseases, making it a reliable tool for disease detection.

The precision of the proposed model was recorded at 99.677%, significantly higher than CNN (92.538%), MobileNetV2 (97.687%), and GoogleNet (94.012%). High precision means that when the model predicts a plant has a certain disease, it is very likely to be correct, minimizing false positive results. Similarly, the recall score for the proposed model was the highest at 99.648% compared to CNN (92.751%), MobileNetV2 (97.626%), and GoogleNet (97.568%). A high recall signifies that the model is effective at identifying the majority of actual disease cases, reducing the number of false negatives.

The F1-score, a measure that combines precision and recall, was also the highest for the proposed model at 99.654%, indicating a balanced performance between precision and recall. In contrast, the F1-scores for CNN, MobileNetV2, and GoogleNet were 92.389%, 97.357%, and 97.254% respectively. A high F1-score further emphasizes the model's ability to strike a balance between minimizing both false positives and false negatives, crucial for real-world plant disease detection applications.

Referencing Table 3, it's evident that the proposed model was trained for 20 epochs, outperforming the other models in terms of accuracy, precision, recall, and F1-score. These results affirm the robust predictive capabilities of the proposed model for image classification tasks.

5.2.2 Visualization with Grad-CAM

The red areas on the heatmap indicate the regions with the highest importance for the model’s prediction. The yellow and green areas show regions of moderate importance. The blue areas represent regions with the least importance.

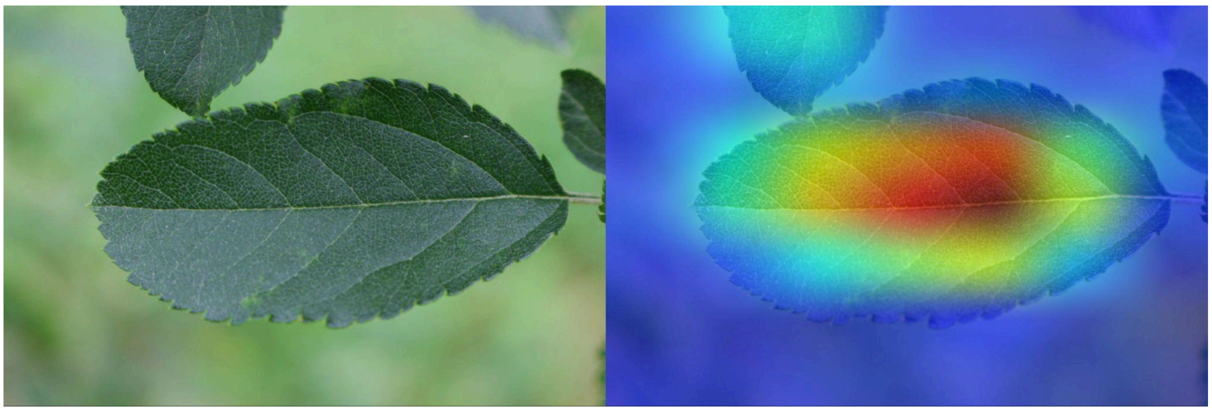

Figure 10. Healthy Leaf

For a healthy leaf in Figure 10. The model might focus on the texture, color, and vein patterns to determine its health status. The heatmap would likely highlight the central part of the leaf where these features are most prominent. The gradations of color from red to blue would visually represent how the importance decreases from the center towards the edges of the leaf.

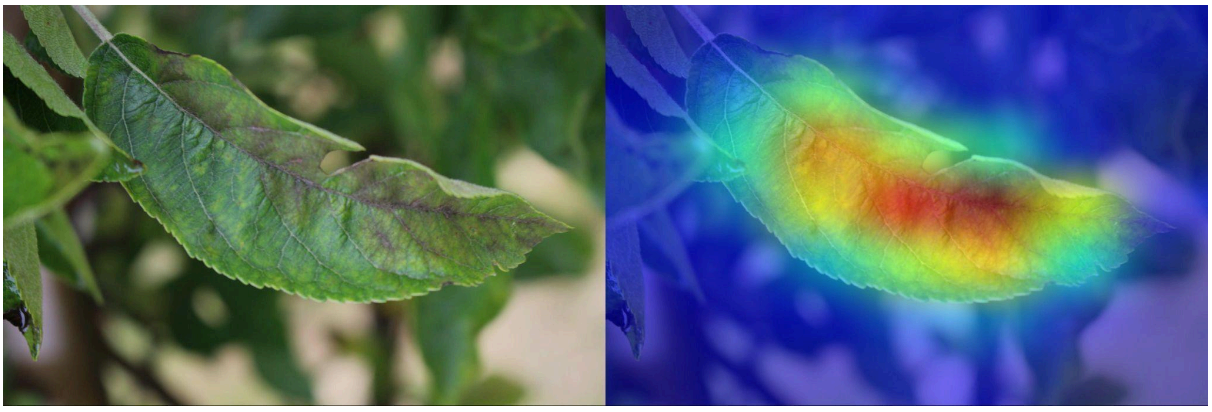

Figure 11. Apple Scrap Leaf

The leaf in Figure 11 shows classic symptoms of apple scab disease, characterized by dark spots and discoloration. These lesions are the result of the fungus Venturia inaequalis, which infects the leaves and fruit of apple trees. The right side of the image, which seems to be a specialized imaging representation, displays a range of colors from blue to red/yellow. This could indicate varying conditions on the leaf's surface, with warmer colors highlighting the areas most affected by the disease.

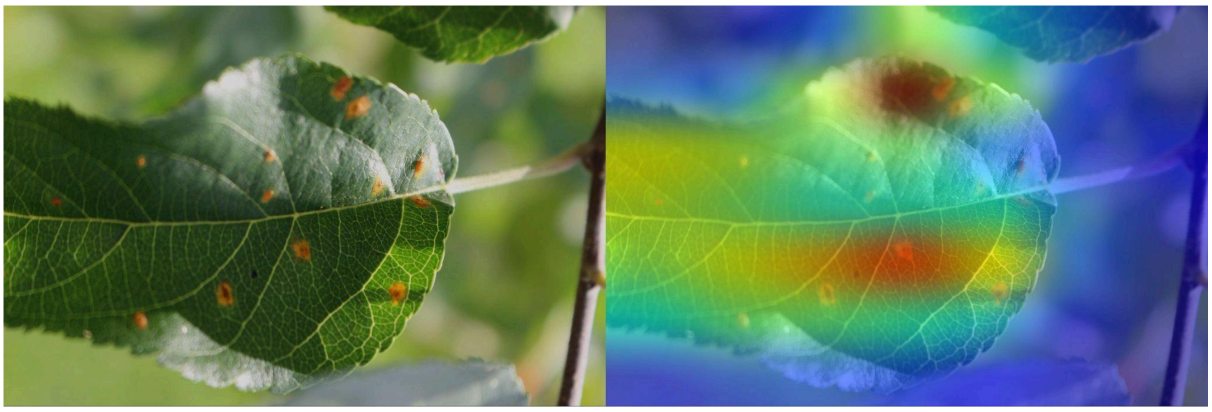

Figure 12. Rust Leaf

Figure 12, similar to Figure 11, shows a Grad-CAM visualization. The warmer colors in the heatmap likely indicate areas where the rust leaf disease is most severe. Conversely, the cooler colors likely represent healthier parts of the leaf.

In the context of the proposed model, which leverages the EfficientNetB5 architecture, Grad-CAM serves as an instrumental tool for visualizing the decision-making process of the model. By generating heatmaps, Grad-CAM illuminates the regions of the image that the model deems significant for its predictions. This visualization provides a deeper understanding of the model's interpretative process and validates its predictions.

For the specific task of plant disease detection, Grad-CAM visualizations can confirm whether the model is focusing on the correct features, such as the disease symptoms on the leaves. If the model is well-trained, the heatmap should ideally highlight areas where disease symptoms are present, thereby providing a visual confirmation of the model's focus and accuracy. If the heatmap highlights regions that are not relevant to the disease symptoms, it could indicate that the model is being influenced by noise or irrelevant features in the data.

In conclusion, Grad-CAM emerges as a powerful tool for understanding, validating, and improving the model's performance. It provides a more intuitive way to interpret the model's predictions, complementing other evaluation metrics. However, it is crucial to use Grad-CAM judiciously and in conjunction with other evaluation methods to ensure a comprehensive assessment of the model's performance. This study underscores the importance of such visualization tools in the broader context of deep learning and their potential to enhance the interpretability and reliability of models.

6 Conclusion and Future Work

6.1 Contributions of the Study

The research conducted has led to significant findings in the field of plant disease detection. The study emphasized the effectiveness of the EfficientNet model, which demonstrated superior performance across all key metrics on the PlantVillage dataset. The model achieved an accuracy of 99.66%, outperforming other models such as CNN, MobileNetV2, and GoogleNet.

The use of Grad-CAM contributed to the interpretability of the model, providing visual explanations for the model's predictions. This visualization technique allowed for a deeper understanding of the model's decision-making process, highlighting the areas of the image that the model deemed most important for making its predictions.

This study makes specific contributions to the field of plant disease detection. It demonstrates the application of the EfficientNet model for plant disease classification, achieving high accuracy and efficiency. The integration of Grad-CAM provides a means to visualize and understand the model's predictions, enhancing the interpretability of the model.

These contributions could potentially advance the development of more accurate and efficient plant disease detection systems. The findings of this study could serve as a foundation for future research in this field, paving the way for the development of advanced disease detection systems that can help farmers and agricultural industries improve crop yield and reduce costs associated with disease management.

6.2 Implications and Applications

The developed system has potential real-world applications in agriculture and other fields. It can be used for early detection of plant diseases, enabling timely intervention and treatment to prevent crop losses. This could significantly benefit farmers and agricultural industries by improving crop yield and reducing costs associated with disease management.

The system could also be integrated into mobile applications, providing farmers with a convenient and accessible tool for plant disease detection. This could revolutionize the way farmers monitor the health of their crops, allowing them to detect and treat diseases early, thereby minimizing crop losses and maximizing yield.

6.3 Recommendations for Future Research

Future research could explore strategies to reduce the training time of the EfficientNet model without compromising its performance. Additionally, the model could be tested on other plant disease datasets to evaluate its generalizability. Further research could also investigate the integration of other visualization techniques to enhance the interpretability of the model.

There is also scope for exploring the application of the model in other domains. For instance, the model could be adapted for disease detection in other types of plants or even for detecting diseases in animals. This would require additional research to adapt the model to these new tasks and to gather and preprocess the necessary data.

6.4 Conclusion

In conclusion, this research has demonstrated the effectiveness of the EfficientNet model for plant disease detection. The study has shown that the model, coupled with Grad-CAM for visualization, can achieve high accuracy and efficiency. This research contributes to the advancement of plant disease detection systems, with potential real-world applications in agriculture and beyond.

The findings of this study underscore the importance of leveraging advanced deep learning models and visualization techniques in the field of plant disease detection. They highlight the potential of these technologies to revolutionize disease detection in agriculture, leading to improved crop yields and reduced costs. The research also opens up new avenues for future research in this field, setting the stage for further advancements in plant disease detection.

References

[1] “ Plant Production and Protection.” n.d. Food and Agriculture Organization of the United Nations. https://www.fao.org/plant-production-protection/about/en.

[2] “About Python.” n.d. Python Institute. https://pythoninstitute.org/about-python.

[3] Choi, Yoojin, Jihwan Choi, Mostafa El-Khamy, and Jungwon Lee. 2020. “Data-Free Network Quantization With Adversarial Knowledge Distillation.” arXiv. https://arxiv.org/abs/2005.04136.

[4] Fu, Yixing. 2020. “Image classification via fine-tuning with EfficientNet.” Keras. https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/#transfer-learning-from-pretrained-weights.

[5] Ganesh, Prakhar. 2020. “From LeNet to EfficientNet: The evolution of CNNs.” Towards Data Science. https://towardsdatascience.com/from-lenet-to-efficientnet-the-evolution-of-cnns-3a57eb34672f.

[6] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. N.p.: MIT Press. https://www.deeplearningbook.org/.

[7] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. “Deep Residual Learning for Image Recognition.” arXiv. https://arxiv.org/abs/1512.03385.

[8] Howard, Andrew G., Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.” arXiv. https://arxiv.org/abs/1704.04861.

[9] “Introduction to NumPy.” n.d. W3Schools. Accessed May 23, 2024. https://www.w3schools.com/python/numpy/numpy_intro.asp.

[10] Jain, Sandeep. 2024. “Python Numpy.” GeeksforGeeks. https://www.geeksforgeeks.org/python-numpy/.

[11] Jain, Sandeep. 2024. “Introduction to Matplotlib.” GeeksforGeeks. https://www.geeksforgeeks.org/python-introduction-matplotlib/.

[12] Jain, Sandeep. 2024. “What is Keras?” GeeksforGeeks. https://www.geeksforgeeks.org/what-is-keras/.

[13] K., Thenmozhi, and U. Srinivasulu Reddy. n.d. "Crop pest classification based on deep convolutional neural networks and transfer learning." https://www.sciencedirect.com/science/article/abs/pii/S0168169919310695?via%3Dihub.

[14] Mengistu, Abrham Debasu, Seffi Gebeyehu Mengistu, and Dagnachew Melesew. n.d. "An Automatic Coffee Plant Diseases Identification Using Hybrid Approaches of Image Processing and Decision Tree." ResearchGate. https://www.researchgate.net/publication/322616034_An_Automatic_Coffee_Plant_Diseases_Identification_Using_Hybrid_Approaches_of_Image_Processing_and_Decision_Tree.

[15] Mohanty, Sharada. n.d. "PlantVillage-Dataset: Dataset of diseased plant leaf images and corresponding labels." GitHub. Accessed May 23, 2024. https://github.com/spMohanty/PlantVillage-Dataset.

[16] "Package overview — pandas 2.2.2 documentation." n.d. Pandas. Accessed May 23, 2024. https://pandas.pydata.org/pandas-docs/stable/getting_started/overview.html.

[17] "Pandas Introduction." n.d. W3Schools. Accessed May 23, 2024. https://www.w3schools.com/python/pandas/pandas_intro.asp.

[18] "Python Imaging Library." n.d. Wikipedia. Accessed May 23, 2024. https://en.wikipedia.org/wiki/Python_Imaging_Library.

[19] "Researchers Helping Protect Crops From Pests | NIFA." 2023. USDA NIFA. https://www.nifa.usda.gov/about-nifa/blogs/researchers-helping-protect-crops-pests.

[20] "Scikit Learn Tutorial." n.d. Tutorialspoint. Accessed May 23, 2024. https://www.tutorialspoint.com/scikit_learn/index.htm.

[21] Szegedy, Christian Szegedy. 2014. "Going Deeper with Convolutions." arXiv. https://arxiv.org/abs/1409.4842.

[22] Tan, Mingxing, and Quoc V. . 2019. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." arXiv. https://arxiv.org/abs/1905.11946.

[23] "TensorFlow." n.d. Wikipedia. Accessed May 23, 2024. https://en.wikipedia.org/wiki/TensorFlow.

[24] Picon A, Seitz M, Alvarez-Gila A, Mohnke P, Ortiz-Barredo A, Echazarra J. “Crop conditional convolutional neural networks for massive multi-crop plant disease classification over cell phone acquired images taken on real field conditions.” Comput Electron Agric. 2019;167:105093. https://doi.org/10.1016/j.compag.2019.105093.

[25] Fang T, Chen P, Zhang J, Wang B. Crop leaf disease grade identification based on an improved convolutional neural network. J Electron Imaging. 2020;29(1):1. https://doi.org/10.1117%2F1.JEI.29.1.013004.

[26] Nagasubramanian K, Jones S, Singh AK, Sarkar S, Singh A, Ganapathysubramanian B. Plant disease identification using explainable 3D deep learning on hyperspectral images. Plant Methods. 2019;15(1):1–10. https://link.springer.com/article/10.1186/s13007-019-0479-8.

[27] Tianjiao C, Wei D, Juan Z, Chengjun X, Rujing W, Wancai L, et al. Intelligent identification system of disease and insect pests based on deep learning. China Plant Prot Guide. 2019;039(004):26–34.

[28] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, Dhruv Batra. 2019. “Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization.” arXiv. https://arxiv.org/abs/1610.02391.

[29] Fuentes A, Yoon S, Kim SC, Park DS. A robust deep-learning-based detector for real-time tomato plant diseases and pests detection. Sensors. 2017;17(9):2022.

[30] Ozguven MM, Adem K. Automatic detection and classification of leaf spot disease in sugar beet using deep learning algorithms. Phys A Statal Mech Appl. 2019;535(2019):122537.

[31] Xie X, Ma Y, Liu B, He J, Wang H. A deep-learning-based real-time detector for grape leaf diseases using improved convolutional neural networks. Front Plant Sci. 2020;11:751.

[32] Bhatt PV, Sarangi S, Pappula S. Detection of diseases and pests on images captured in uncontrolled conditions from tea plantations. In: Proc. SPIE 11008, autonomous air and ground sensing systems for agricultural optimization and phenotyping IV; 2019. p. 1100808. https://doi.org/10.1117/12.2518868.

[33] Zhang, B., Zhang, M., & Chen, Y. (2019). Crop pest identification based on spatial pyramid pooling and deep convolutional neural network. Trans Chin Soc Agric Eng, 35(19), 209-15.

Turnitin Originality Report

Processed on: 19-Jun-2024 09:40 +07
ID: 2405102637
Word Count: 14566
Submitted: 1

Graduation Thesis By Hoàng Diệu Linh

Similarity Index	Similarity by Source	
10%	Internet Sources:	8%
	Publications:	10%
	Student Papers:	7%

2% match (student papers from 01-Jun-2023)
Submitted to National Economics University on 2023-06-01

1% match (Phuc Nguyen, Thang Truong, Nguyen D. Vo, Khang Nguyen. "Rethinking Classification of Oriented Object Detection in Aerial Images", International Journal of Advanced Computer Science and Applications, 2022)
Phuc Nguyen, Thang Truong, Nguyen D. Vo, Khang Nguyen. "Rethinking Classification of Oriented Object Detection in Aerial Images", International Journal of Advanced Computer Science and Applications, 2022

1% match (Internet from 25-Jul-2021)
https://plantmethods.biomedcentral.com/track/pdf/10.1186/s13007-021-00722-9.pdf

1% match (Internet from 10-Jun-2024)
https://www.fastercapital.com/keyword/f1-score.html

1% match (Internet from 01-Dec-2023)
https://wbchse.wb.gov.in/wp-content/uploads/2023/10/AIDS_XI_SEC2.pdf

1% match (Internet from 27-Nov-2022)
https://cse.anits.edu.in/projects/projects2021A5.pdf

1% match (Internet from 14-Mar-2024)
https://dergipark.org.tr/tr/download/article-file/3693293

1% match (Internet from 03-Oct-2023)
https://www.engpaper.com/download/book-recommender-system.pdf

< 1% match (student papers from 06-Jun-2024)
Submitted to Liverpool John Moores University on 2024-06-06

< 1% match (Internet from 02-Apr-2023)
https://zuscholars.zu.ac.ae/cgi/viewcontent.cgi?article=6755&context=works

< 1% match (Internet from 14-May-2024)
http://fastercapital.com/keyword/recall-score.html

< 1% match (Internet from 09-Jun-2017)
http://www.ripublication.com/ijaer16/ijaerv11n12_03.pdf

< 1% match (student papers from 24-Oct-2007)