Faculty of Electrical Technology and Engineering

PLANT DISEASE DETECTION USING CONVOLUTIONAL NEURAL NETWORK

MUHAMMAD FAZA IQMAL BIN MUSTAFA KAMAL

Bachelor of Electrical Engineering Technology (Industrial Automation & Robotics) with Honours

2024

PLANT DISEASE DETECTION USING CONVOLUTIONAL NEURAL NETWORK

MUHAMMAD FAZA IQMAL BIN MUSTAFA KAMAL

A project report submitted
in partial fulfillment of the requirements for the degree of
Bachelor of Electrical Engineering Technology (Industrial Automation & Robotics)
with Honours

Faculty of Electrical Technology and Engineering

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2024

DECLARATION

I declare that this project report entitled “Plant Disease Detection Using Convolutional Neural Network” is the result of my own research except as cited in the references. The project report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature	:	
Student Name	:	MUHAMMAD FAZA IQMAL BIN MUSTAFA KAMAL
Date	:	5/1/2025

DEDICATION

Dedicated to the journey of my final year project — a labour of passion, dedication, and countless hours of hard work. To my loving parents, Mustafa kamal Bin Abdul Rahim and Zulaidah Binti Che Abdul Halim, your unwavering support and belief in my abilities have been my driving force.

Special gratitude to my exceptional supervisor, Ts. Aminurrashid Bin Noordin and my co-supervisor, Ts. Madiha Binti zahari for your invaluable guidance, encouragement, and expertise. Your mentorship has played a pivotal role in shaping the course of this project.

To my dear friends, who have shared both the challenges and triumphs of this endeavour, thank you for your camaraderie and understanding.

This work is dedicated to all those who find joy in the pursuit of knowledge and the quest for excellence. May it stand as a testament to the collective effort and passion that fuelled this final year project. Thank you for being part of this significant chapter in my academic journey.

ABSTRACT

This project focuses on developing a plant disease detection system using Convolutional Neural Networks (CNN) to address the critical challenge of identifying plant diseases early in agriculture. The proposed system leverages image analysis to classify diseases such as Grape Black Rot, Leaf Blight, and healthy conditions in grape leaves. Utilizing AlexNet architecture in MATLAB, the model processes a dataset of 500 leaf images (70% for training, 30% for testing) with image preprocessing techniques like resizing and normalization.The methodology involves designing a MATLAB-based GUI for user interaction, allowing image uploads, disease detection, affected area analysis, and remedy suggestions. Model performance was evaluated on multiple metrics, achieving an overall accuracy of 99.3% on the validation dataset. Tests on 60 samples consistently demonstrated high prediction confidence (96.42%-100%) and accurate classification of healthy and diseased leaves.Quantitative analysis of the affected area using clustering revealed detailed insights into disease severity, supporting effective decision-making. This system shows strong potential for real-time agricultural applications, contributing to sustainable farming practices and enhancing food security. Future enhancements include integrating mobile platforms for broader accessibility.

ABSTRAK

Projek ini bertujuan untuk membangunkan sistem pengesanan penyakit tumbuhan menggunakan Rangkaian Neural Konvolusi (CNN) bagi menangani cabaran kritikal dalam mengenal pasti penyakit tumbuhan secara awal dalam sektor pertanian. Sistem yang dicadangkan memanfaatkan analisis imej untuk mengelaskan penyakit seperti Grape Black Rot, Leaf Blight, dan keadaan sihat pada daun anggur. Dengan menggunakan seni bina AlexNet dalam MATLAB, model ini memproses dataset yang terdiri daripada 500 imej daun (70% untuk latihan, 30% untuk ujian) dengan teknik prapemprosesan imej seperti pengubahan saiz dan penormalan. Metodologi melibatkan reka bentuk GUI berasaskan MATLAB untuk interaksi pengguna, membolehkan muat naik imej, pengesanan penyakit, analisis kawasan terjejas, dan cadangan rawatan. Prestasi model dinilai berdasarkan pelbagai metrik, mencapai ketepatan keseluruhan sebanyak 99.3% pada dataset validasi. Ujian ke atas 60 sampel menunjukkan keyakinan ramalan yang tinggi (96.42%-100%) dan pengelasan yang tepat antara daun yang sihat dan yang berpenyakit. Analisis kuantitatif kawasan terjejas menggunakan pengelompokan memberikan maklumat terperinci tentang tahap keterukan penyakit, menyokong pembuatan keputusan yang berkesan. Sistem ini menunjukkan potensi yang kuat untuk aplikasi masa nyata dalam pertanian, menyumbang kepada amalan pertanian mampan dan meningkatkan keselamatan makanan. Penambahbaikan masa depan termasuk integrasi platform mudah alih untuk akses yang lebih luas.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratitude to my supervisor, Ts. Aminurrashid Bin Noordin and co-supervisor, Ts. Madiha Binti Zahari for their precious guidance, words of wisdom and patient throughout this project.

I am also indebted to Universiti Teknikal Malaysia Melaka (UTeM) for the financial support through which enables me to accomplish the project. Not forgetting my fellow colleagues, for the willingness of sharing their thoughts and ideas regarding the project.

My highest appreciation goes to my parents, parents' in-law, and family members for their love and prayer during the period of my study. An honourable mention also goes to my academic advisor, for all the motivation and understanding.

Finally, I would like to thank all the staffs at the Universiti Teknikal Malaysia Melaka (UTeM), fellow colleagues and classmates, the faculty members, as well as other individuals who are not listed here for being co-operative and helpful.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

δ	-	Voltage angle
	-	
	-	
	-	
	-	
	-	
	-	
	-	

LIST OF ABBREVIATIONS

V	-	Voltage
	-	
	-	
	-	
	-	
	-	
	-	
	-	

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

Agriculture is the cornerstone of human civilization, providing essential resources such as food, fiber, and fuel. It encompasses a broad range of activities, including the cultivation of plants and the rearing of animals, which are fundamental to sustaining the global population. With the ever-growing demand for agricultural products driven by population growth and changing dietary patterns, ensuring the health and productivity of crops is paramount.

One of the most significant challenges in agriculture is the management of plant diseases. Plant diseases, caused by various pathogens including bacteria, viruses, fungi, and nematodes, can lead to substantial losses in crop yield and quality. These diseases can spread rapidly through fields and across regions, causing epidemics that threaten food security and economic stability. Effective disease management is thus crucial for maintaining high agricultural productivity and ensuring a stable food supply.

Traditionally, plant disease management has relied on manual inspection and the application of chemical treatments. However, these methods have limitations. Manual inspection is time-consuming, labor-intensive, and prone to human error, while the excessive use of chemical treatments can lead to environmental pollution, resistance development in pathogens, and increased production costs.

In recent years, advancements in technology have opened new avenues for plant disease detection and management. Precision agriculture, leveraging tools such as remote

sensing, machine learning, and computer vision, offers innovative solutions to these challenges. By automating the detection and diagnosis of plant diseases, these technologies can enhance the accuracy and efficiency of disease management practices.

Machine learning and computer vision have shown great promise in identifying and classifying plant diseases based on visual symptoms. These technologies analyze images of plants to detect early signs of disease, enabling timely intervention and minimizing crop losses. Moreover, the integration of such technologies into user-friendly interfaces, such as graphical user interfaces (GUIs), makes these tools accessible to farmers and agricultural professionals, facilitating widespread adoption.

This project focuses on leveraging machine learning and MATLAB to develop a system for detecting plant diseases through image analysis. By capturing and processing images of plants, the system aims to accurately identify the presence of diseases and provide reliable results through an intuitive GUI. This approach not only enhances the precision of disease detection but also contributes to the overall goal of sustainable agriculture by promoting efficient and environmentally friendly disease management practices.

1.2 Problem Statement

The early and accurate detection of plant diseases is a critical challenge in agriculture, affecting crop yields, food security, and economic stability. Traditional methods of disease detection, which often rely on manual inspection by trained experts, are time-consuming, labor-intensive, and prone to human error. Additionally, these methods are not scalable for large-scale farming operations and may result in delayed intervention, leading to significant crop losses. There is an urgent need for an efficient, scalable, and reliable solution that can identify plant diseases at an early stage to enable timely and effective management.

This problem directly impacts Sustainable Development Goal (SDG) 2: Zero Hunger, which aims to end hunger, achieve food security, improve nutrition, and promote sustainable agriculture. By addressing the challenge of plant disease detection, we can significantly reduce crop losses and improve agricultural productivity, thereby contributing to food security and sustainable farming practices.

To tackle these challenges, this research proposes the development of a smart plant disease detection system using Convolutional Neural Networks (CNNs), leveraging their superior image processing capabilities to identify a wide array of plant diseases automatically and accurately from leaf images. This system aims to provide a practical, real-time solution for farmers and agricultural professionals, enhancing disease management practices and contributing to the achievement of SDG 2 by ensuring a more reliable and sustainable food supply.

1.3 Project Objective

The objectives of this project are as follows:

a) Develop a smart plant disease detection using Convolutional Neural Network (CNNs).

b) To collect and create a Comprehensive Dataset: Gather a diverse set of high-quality images of plant leaves affected by various diseases, as well as healthy leaves, across different plant species.

c) To analyze the performance of the system based on the accuracy and efficiency of the system.

1.4 Scope of Project

The scope of this project are as follows:

a) Investigation of load profiles of residential, commercial and industrial load segments to determine load factor (LF) and loss factor (LsF) were considered in the analytical models. The model will focus on detecting and classifying common leaf-based plant diseases from a single plant species which is grape leaf.

b) This project use 500 datset, 70% for training and 30% for testing.

c) The dataset contains three differences categories which are ‘Healthy leaf’, ‘Grape Black Rot’ and ‘Leaf Blight’.

d) Image preprocessing will include resizing, normalization, and basic augmentation to ensure consistent input data quality for the CNN model.

e) The model’s performance will be evaluated using standard metrics (accuracy) on a test dataset derived from the training dataset's domain.

f) The CNN will be optimized for MATLAB to run efficiently on standard desktop systems, targeting inference times under 1 second per image.

g) A basic MATLAB GUI will be developed to allow users to upload images and receive predictions with disease classification, confidence scores, percentage of Affection and remedy of diseases.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The literature review section of this research project delves into a comprehensive analysis of scholarly journals, product research studies, and technological advancements relevant to the field of plant disease detection using Convolutional Neural Networks (CNNs). By examining a range of academic sources, including peer-reviewed journals and research articles, this review aims to provide a thorough understanding of the current state of knowledge in the application of CNNs for plant disease identification. Additionally, insights from product research studies focusing on innovative solutions for agricultural disease management will be explored to identify trends, challenges, and opportunities in the field. Furthermore, the review will highlight key technological developments, such as advancements in image processing algorithms and deep learning techniques, that have contributed to the evolution of plant disease detection systems. By synthesizing information from these diverse sources, this literature review sets the stage for the subsequent sections of the research project, guiding the reader through a comprehensive overview of the theoretical and practical aspects shaping the landscape of plant disease detection technologies. Figure 2.1 shows a various healthy plant and the differences disease.

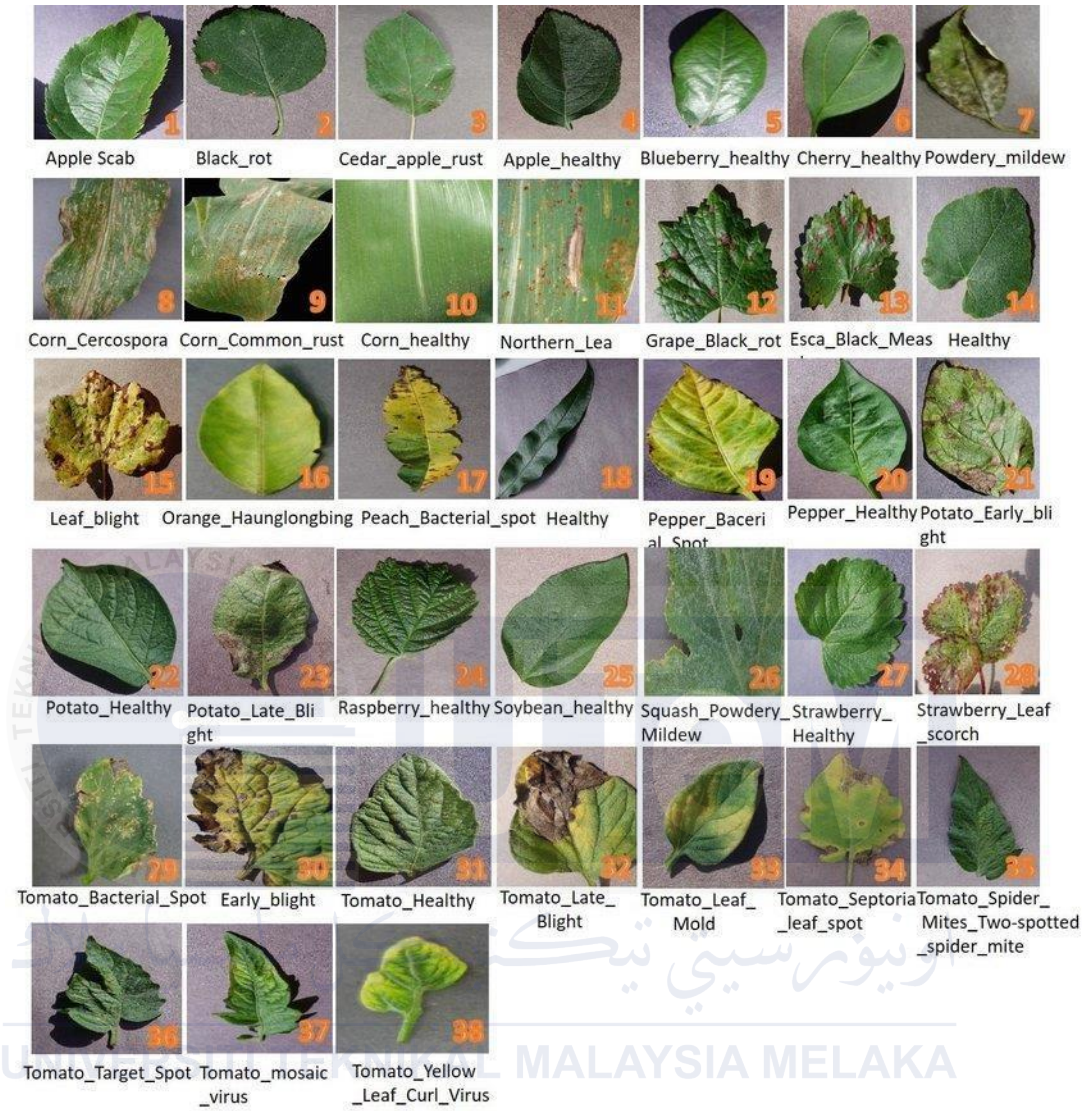


Figure 2.1 Example of various plant disease [1]

2.2 Recent Study

There is much research that had been published that involved in plant disease detection like the work from the author named Sethi at el. presents a novel approach for detecting plant diseases using image segmentation techniques. The study utilizes deep learning algorithms to segment images into healthy and infected areas, enabling the classification of diseases based on the segmented regions. By automating the detection and quantification of plant diseases, the proposed method proves to be a valuable tool for farmers and researchers. Experimental results demonstrate high accuracy in detecting various plant

diseases such as leaf spots, powdery mildew, and rust. Leveraging transfer learning and semantic segmentation, the research showcases the effectiveness of image segmentation in improving the accuracy and precision of disease classification. The study emphasizes the potential of computer vision techniques in enhancing crop management practices and reducing losses due to disease outbreaks [2].

The work by author Srivastava et al. explores the application of deep convolutional networks in developing a model for accurate and efficient plant disease recognition. By utilizing image processing techniques, the study aims to enhance agricultural practices by providing a fast and reliable method for detecting diseases in plants. The research emphasizes the importance of timely and precise diagnosis in ensuring sustainable agriculture and combating the development of pathogen resistance due to indiscriminate pesticide use. By deep learning frameworks and fine-tuning parameters, the study achieved an overall accuracy of 88% in disease classification. The methodology involves gathering and preprocessing leaf images, training the deep CNN, and conducting tests to evaluate the model's performance. The findings suggest a promising approach for automating disease detection in plants, with potential applications in real-time crop surveillance and yield prediction. The study also mentions that the system is based on Python and can provide an accuracy of around 88%, which can be further optimized by using Google's GPU for processing [3].

The work by Dhiman et al. proposed the Smart Disease Detection System for Citrus Fruits Using Deep Learning with Edge Computing to detect diseases in citrus fruits through advanced technology. By combining deep learning methods, specifically a CNN- LSTM model, with edge computing, the system aims to enhance the efficiency and accuracy of disease classification. The research addresses the growing need for automated fruit disease detection to ensure the production of high-quality, disease-free citrus fruits. With the

increasing global production of citrus fruits, such technological advancements are crucial for farmers to maintain quality standards and prevent the spread of diseases.

The proposed system leverages edge computing servers located closer to client devices, reducing data transmission delays, and improving overall performance compared to traditional cloud-based solutions. By integrating cutting-edge technologies, this smart disease detection system offers a promising solution for the agricultural sector to enhance sustainability and productivity in citrus fruit cultivation. In the experiment, the accuracy achieved using the baseline model without using the compression technique is 96.93%. The proposed CNN- LSTM model without compression technique achieved an accuracy of 98.87%. Additionally, the model achieved high accuracies for different classes of citrus fruit diseases, such as 97.08% for Canker, 97.32% for Scab, 94.65% for Melanosis, 95.13% for Greening, 95.38% for Blackspot, and 99.05% for Healthy [4].

The work by Bouni et al. the application of pretrained deep neural networks in predicting tomato leaf diseases. By utilizing transfer learning with models such as AlexNet, VGG16, ResNet, and DenseNet, the study achieved significant accuracy rates, with DenseNet and the RmsProp optimization method leading to the highest accuracy of 99.9%. The research emphasizes the importance of technology, particularly deep learning, and automation, in agriculture to address the challenges of plant disease management. With the global population projected to reach 9.2 billion by 2050, reducing yield losses due to factors like climate change and plant diseases is crucial for food security. The study highlights the potential of advanced technologies in agriculture to enhance disease detection and control, ultimately contributing to sustainable food production and societal well-being.In the experiment, the classification accuracy of the models varied. The accuracy percentages for each model are AlexNet: 85.6%, RmsProp optimizer: 95.4%, VGG-16: RmsProp optimizer: 89.5%, ResNet:Adam optimizer: 91.6%, RmsProp optimizer: 99.2%. DenseNet: Adam

optimizer: 95.6% and RmsProp optimizer: 99.5%. These accuracy percentages reflect the performance of the models in classifying tomato leaf diseases in the experiment conducted by Mohamed Bouni et al [5].

The work by Zamani at el. presented a comprehensive study on utilizing machine learning and image processing techniques to evaluate infected leaf disease images. The research aimed to develop an automated disease detection system for plant leaves, essential for expediting crop diagnosis in agriculture. The process involved multiple stages, including image acquisition, preprocessing to eliminate noise, segmentation using the K-Means approach for boundary establishment, and feature extraction through principal component analysis. Various classification techniques such as RBF-SVM, SVM, random forest, and ID3 were applied for categorizing images based on disease detection. Despite the promising methodology, the article was retracted due to evidence of systematic manipulation in the publication process, casting doubt on the reliability of its content and research findings. This retraction highlights the critical importance of upholding research integrity and verifying the credibility of scientific publications to ensure the dissemination of accurate and trustworthy information in academic circles [6]. The accuracy results for the algorithms ID3, RandomForest, SVM, and RBF SVM were ID3: Approximately 0.94 accuracy, RandomForest: Approximately 0.96 accuracy, SVM: Approximately 0.88 accuracy and RBF SVM: Approximately 0.98 accuracy.

The journal by authors Shewale at el. proposed a high-performance deep learning architecture for early detection and classification of plant leaf disease to address the significant annual production loss caused by plant leaf diseases. By leveraging deep learning techniques, the research focuses on early detection and accurate classification of plant diseases, particularly in tomato plants. The study emphasizes the importance of automated intelligent strategies in reducing manual recognition efforts and improving recognition

accuracy. By combining deep learning with image processing, the proposed method can classify diseases with high precision by automatically extracting features from leaf images captured in real-time agricultural environments. The research aims to provide a clear pathway for crop disease diagnosis on a global scale by training deep learning models on progressively larger and publicly available image datasets. The goal is to develop a practical tool that can assist farmers in efficiently diagnosing plant diseases, leading to more effective decision-making and sustainable agricultural practices. The experiment achieved an accurate percentage of 99.81% with the proposed method. This high accuracy rate demonstrates the effectiveness of the deep learning model in accurately detecting and classifying plant leaf diseases [7].

The work by Amin at el. presents a novel approach to automating the identification and classification of corn leaf diseases using deep learning techniques. The rapid and accurate detection of plant diseases is crucial for global food security, but it remains a challenging and time-consuming task. The proposed model leverages two pre-trained CNNs, EfficientNetB0 and DenseNet121, to extract features from digital images of corn leaves infected with gray leaf spot, common rust, northern leaf blight, and healthy leaves. By utilizing feature fusion techniques between the two CNNs, the model enhances its predictive power and builds an end-to-end classification system. This innovative approach not only addresses the limitations of large parameter sizes in traditional models but also demonstrates the potential of deep learning algorithms in revolutionizing plant disease diagnosis and management in agriculture. The proposed end- to-end deep learning model achieved a classification accuracy of 98.56% in the experiment. This high accuracy rate demonstrates the effectiveness of using deep learning techniques for corn leaf disease classification. The model outperformed other methods and CNNs used in the study, showcasing its robustness and efficiency in identifying different classes of corn leaf diseases [8].

The journal by authors Moupojou presents a FieldPlant, A Dataset of Field Plant Images for Plant Disease Detection and Classification with Deep Learning. The dataset comprises 5,170 original images captured in plantations under various lighting conditions, annotated with 8,629 individual leaf annotations across 27 disease classes. By leveraging the RoboFlow online platform for annotation, researchers can develop models to aid farmers in real-time plant disease identification and classification. The article highlights the limitations of existing datasets, emphasizing the need for more comprehensive and accurate annotations to train high-accuracy models. Through the utilization of deep learning models, the study showcases the potential of automatic feature extraction for improved disease diagnosis. The dataset's creation and potential applications align with addressing global food security challenges by mitigating crop yield losses due to plant diseases. The study by Moupojou et al. achieved top-1 average identification accuracy of 95% on the tomato test dataset using a hybrid deep learning model. Khattak et al. achieved a test accuracy of 94.55% in differentiating healthy citrus fruits and leaves from those with common citrus diseases using a 2-layers CNN model. The study by Moupojou et al.optimized multi-task learning using homoscedastic uncertainty to obtain plant and disease accuracies of 84.71% and 75.06%, respectively, on the PlantDoc dataset [9].

The work by Restrepo-Arias at el. presents a novel approach to plant disease diagnosis using image texture analysis and Bayesian optimization with small neural networks. The method involves preprocessing images to remove background noise, segmenting images into tiles to reduce bias from leaf morphology, and training small convolutional neural network models on a new dataset of images. The proposed strategy aims to avoid classification bias caused by plant characteristics and achieve competitive classification results with efficient computational requirements. Future work includes testing with different image sizes, improving performance metrics, and optimizing other neural

network hyperparameters. The experiment reported in the journal achieved accuracy ranging from approximately 91.50% to 96.31% for different models used in the plant disease detection strategy [10]. The accuracy percentages for each model are MobileNet: 96.31%, SqueezeNet: 95.05%, NasNetMobile: 95.01%, MobileNetV2: 94.59%, ShuffleNet: 91.50%.

The journal by Gao at el explores detecting and identifying potato diseases using multidimensional fusion Atrous-CNN and hyperspectral data. It highlights the importance of accurate disease detection for preserving crop yield and quality, especially concerning diseases like blackleg and soft tuber rot. Traditional methods fall short, prompting the need for advanced technologies. Hyperspectral imaging provides detailed spectral and spatial data crucial for disease detection in potatoes. The study introduces an Atrous-CNN model that combines 1D-CNN, 2D-CNN, and 3D-CNN to enhance disease detection accuracy while reducing hardware consumption. Experimental results show high recognition accuracy, indicating the model's effectiveness in hyperspectral data analysis for potato disease identification. This research not only improves potato disease detection but also has implications for other agricultural crops. By leveraging advanced technologies and innovative network structures, the agricultural industry can enhance disease management strategies, leading to healthier crops and increased productivity. Future work aims to expand this approach to address disease challenges in diverse agricultural settings, promoting precision agriculture and sustainable food production. In the experiment, the Multidimensional Fusion Atrous-CNN model showed high accuracy in identifying potato diseases, with accuracy rates exceeding 99.7%. Compared to other models used, it performed better by improving accuracy by around 0.5% to 0.9%. The 1D-CNN network also achieved high accuracy rates above 98% for disease recognition. Overall, the advanced models proved to be effective in accurately detecting and classifying potato diseases using hyperspectral imaging technology [11].

The work by Neupane at el. explains into the application of Unmanned Aerial Vehicles (UAVs) in precision agriculture for automating the identification and monitoring of plant diseases. By employing UAVs with imaging sensors and machine learning algorithms, farmers can detect biotic injuries caused by pathogens without direct human involvement. Overcoming challenges like limited spectral bandwidth and image noise is addressed through advanced techniques such as transfer learning and batch normalization. Emphasizing the need for extensive datasets, multi-sensor gimbals, and site-specific irradiance systems, the review underscores the potential of UAV technology in revolutionizing plant disease detection and management in agriculture. The integration of UAVs with deep learning methods showcases a promising pathway towards enhancing crop health monitoring and bolstering agricultural productivity. In various, the accuracy percentage in identifying plant diseases using different models ranged from 79% to 99.35% [12].

2.3 Summary Evaluation

The table highlights research on plant disease detection using various methods. Image segmentation achieved 88% accuracy, with plans to improve datasets by adding more disease classes. Convolutional Neural Networks (CNNs) are widely used, with accuracies ranging from 88% to 99.8%, focusing on real-time detection, IoT integration, optimization, and expansion to other plant diseases and parts. Deep Neural Networks (DNNs) showed 83.6% to 99.5% accuracy, with future work aimed at extending to more plant species and gathering diverse datasets. Machine learning methods achieved 88% to 98% accuracy, emphasizing robust models and real-time solutions like mobile apps. Overall, future directions include enhancing datasets, optimizing models, and developing practical, real-time applications for farmers. Refer to table 2.1.

Table 2.1 Comparison of difference method in plant disease detection

Paper	Method	Percentage of accuracy (%)	Future work
[2]	Image Segmentation	88	Dataset Enrichment: The Field Plant dataset can be improved by adding more disease classes, increasing its usefulness for plant disease research and management. Future work should involve collecting more field images to include new disease classes, enhancing the dataset's diversity.
[3]	Convolutional Neural Network	88	1. Automation of Real-Time Detection: Implementing a system to automate the process of detecting yield crops in real-time. 2. Application Development: Creating web or desktop applications to display the prediction results for easy access and interpretation. 3. Optimization for AI Environment: Enhancing the system to operate efficiently in an artificial intelligence (AI) environment.
[4]	Convolutional Neural Network	96.93	1. Optimization Techniques: Further exploration of optimization techniques to enhance the efficiency and speed of the CNN- LSTM model on edge computing devices.Investigate advanced pruning and quantization methods to reduce model size without compromising accuracy. 2. Integration of IoT: Integration of Internet of Things (IoT) devices for real-time data collection and analysis, enabling proactive disease detection and management in citrus orchards.
[5]	Deep Neural Network	83.6-99.5	1. Expansion to Other Plant Species: Extend the research to investigate the application of deep learning models and transfer learning techniques for predicting diseases in a variety of plant species beyond tomatoes. 2. Enhanced Data Collection: Gather more diverse and extensive datasets to train deep learning models effectively, improving the accuracy and generalization of disease prediction across different plant types.

[6]	Machine Learning	88-98	Focus on enhancing the model's accuracy and robustness by incorporating larger, more diverse datasets and employing advanced deep learning techniques such as transfer learning and ensemble methods. Additionally, efforts could be directed towards developing real-time, field- deployable solutions such as mobile apps or handheld devices that integrate the detection model.
[7]	Convolutional Neural Network	99.8	Future Directions: The study proposes developing a server-side system integrated with a handheld tool for farmers, providing real- time disease diagnosis and decision-making. It also suggests expanding the application to recognize diseases in fruits, flowers, and vegetables from leaf images. Future research will focus on covering more plant leaf species, supporting sustainable agricultural development.
[8]	Convolutional Neural Network	98.56	Expansion to Other Plant Diseases: The model's approach can be extended to classify a broader range of plant diseases beyond corn, encompassing various crops and vegetation. This expansion would contribute to the development of a versatile and comprehensive disease detection system for agriculture
[9]	Convolutional Neural Network	95	Dataset Enrichment: The FieldPlant dataset has the potential to be enriched with more disease classes, thereby expanding its scope and utility for plant disease research and management. Future efforts may focus on collecting additional field images to incorporate new disease classes and enhance the dataset's diversity.
[10]	Small Neural Network	91.5-96.31	Experimentation with Different Image Sizes: Conducting tests with various image sizes to enhance resolution for improved classification based on texture features caused by diseases in plant leaves
[11]	Convolutional Neural Network	98	Improving the accuracy and speed of the Atrous-CNN by integrating more advanced deep learning techniques and leveraging larger and more diverse hyperspectral datasets. Another potential direction is to develop a user- friendly mobile application for farmers, enabling real-time disease detection and management in the Field.
[12]	Convolutional Neural Network	79-99.35	Integration of gimbal systems with multiple sensors for enhanced data collection.Expansion of datasets for training and validation to improve algorithm accuracy. Development of site-specific irradiance systems for more precise monitoring.

2.4 Convolutional Neural Network (CNNs)

Artificial Neural Networks (ANN) are algorithmic mathematical models that simulate the composition and operation of biological nervous systems while processing data in distributed parallel. By internally altering the weight relationship between neurons and neurons, ANN accomplishes its goal of processing information. A feedforward neural network made up of numerous convolutional layers and pooling layers is known as a convolutional neural network (CNN). CNN performs very well in multiple tasks, especially in image processing [13].

The concept of CNNs is inspired by the visual processing mechanisms in the mammalian brain, particularly the visual cortex. In the 1960s, David Hubel and Torsten Wiesel conducted pioneering research on the visual cortex of cats [14]. They discovered that individual neurons in the brain's visual cortex respond to specific regions of the visual field, a phenomenon known as receptive fields. This research earned them the Nobel Prize in Physiology or Medicine in 1981 [15]. Figure 2.2 show an example of CNN.

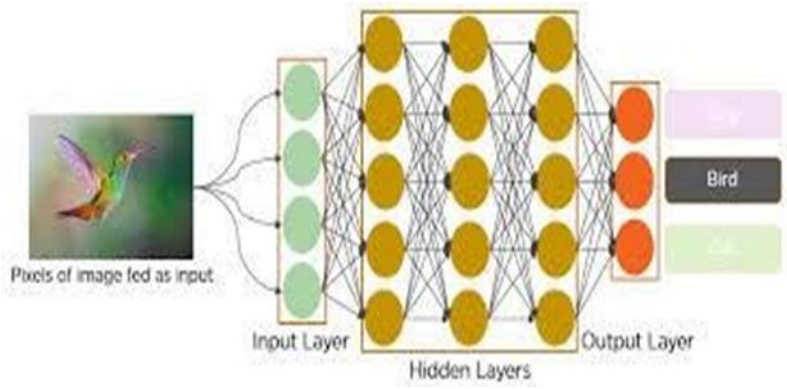


Figure 2.2 Convolutional Neural Networks (CNNs) [15]

In 1980, Kunihiko Fukushima proposed the Neocognition, an early neural network model designed for pattern recognition. It introduced the concept of hierarchical layers with local receptive fields and weight sharing, fundamental ideas that influenced later CNN models [16]. Figure 2.3 shows a Neocognition system.

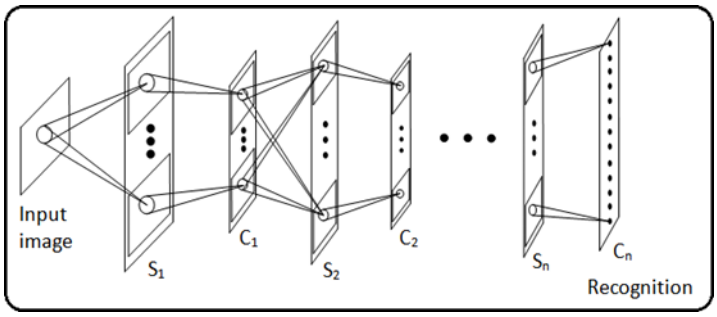


Figure 2.3 Neocognition [15]

In the late 1980s and early 1990s, Yann LeCun and his colleagues developed LeNet-5, a CNN architecture for handwritten digit recognition. LeNet-5 was successfully applied to the MNIST dataset, demonstrating the practical utility of CNNs in image recognition tasks. LeNet-5 incorporated several key components of modern CNNs, including convolutional layers, subsampling (pooling) layers, and fully connected layers. Figure 2.4 shows an example of LeNet-5.

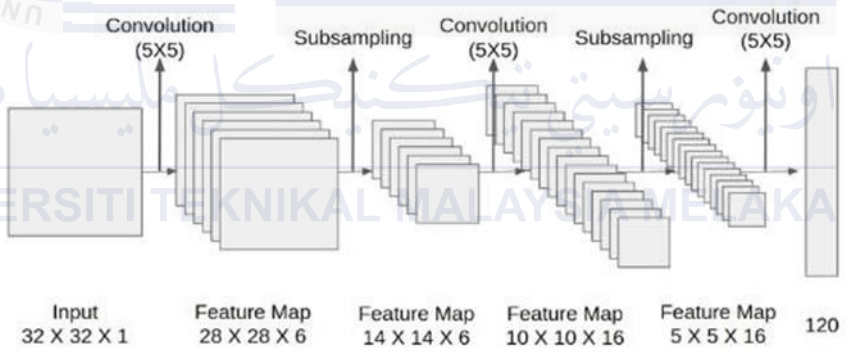


Figure 2.4 LeNet-5 [15]

In 2012, Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton developed AlexNet, a deep CNN that won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) with a significant margin. AlexNet demonstrated the power of deep learning and CNNs in handling large-scale image classification tasks. It popularized the use of ReLU (Rectified Linear Unit) activation functions, dropout for regularization, and GPU acceleration for training large models [15].

In CNNs, a two-stage approach typically refers to a method used in object detection tasks. This approach divides the process into two distinct phases: region proposal and region

classification. The most notable example of this methodology is the Region-based Convolutional Neural Network (R-CNN) family of algorithms, including R-CNN, Fast R-CNN, and Faster R-CNN [17]. Figure 2.5 shows Two- stage approach, Figure 2.6 shows architecture of R-CNN, Figure 2.7 shows architecture of Fast R-CNN, and Figure 2.8 shows architecture of Faster R-CNN.

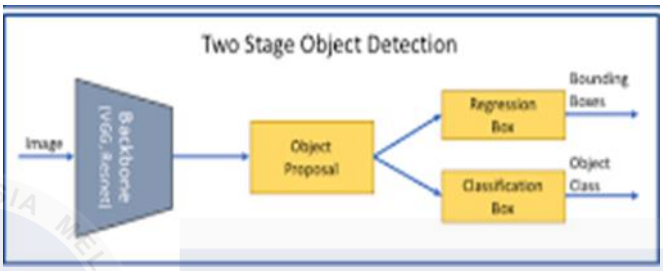


Figure 2.5 Two-stage approach [17]

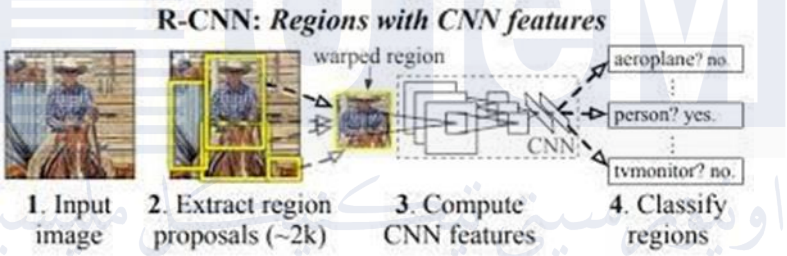


Figure 2.6 Architecture of R-CNN

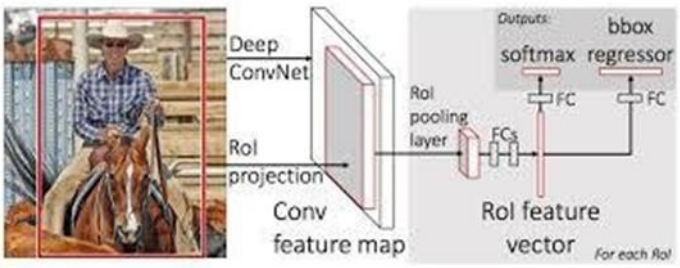


Figure 2.7 Architecture of Fast R-CNN

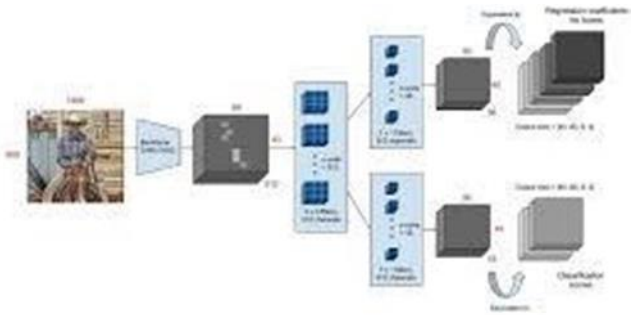


Figure 2.8 Architecture of Faster R-CNN

The one-stage approach in CNNs for object detection simplifies the detection pipeline by combining region proposal and region classification into a single step. This makes the process faster and often more efficient, though sometimes at the expense of accuracy compared to two-stage methods. Key examples of one-stage approaches include YOLO (You Only Look Once) [18] and SSD (Single Shot MultiBox Detector) [19]. Figure 2.9 shows the Architecture of Yolo, Figure 2.10 shows the Architecture of SSD, and Figure 2.11 shows the Architecture of RetinaNet.

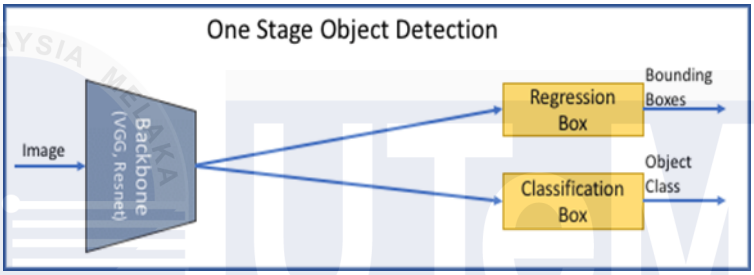


Figure 2.9 Architecture of Yolo [18]

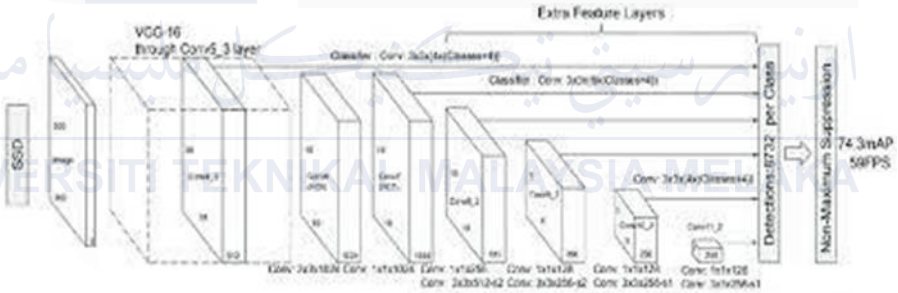


Figure 2.10 Architecture of SSD [19]

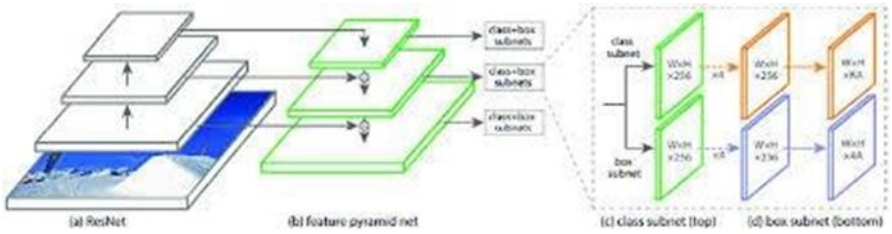


Figure 2.11 Architecture of RetinaNet [19]

2.5 Summary

In summary, based on previous research on suitable computer vision techniques for image processing reveals a variety of traditional methods, such as filtering, edge detection,

and segmentation, which have been fundamental in enhancing and analyzing images. CNNs have emerged as the most suitable and effective technique due to their ability to automatically learn hierarchical features directly from raw images, enabling end-to-end learning that simplifies the pipeline and improves performance. CNNs' robustness, scalability, and superior accuracy in tasks like object detection, recognition, and segmentation make them the preferred choice over traditional and other machine learning-based approaches.

CHAPTER 3

METHODOLOGY

3.1 Introduction

The methodology for this project follows a structured, phased approach to ensure thoroughness and efficiency. Starting with Project Initialization, the foundation is established through defining scope and objectives. This is followed by Requirement Analysis to gather and prioritize all necessary requirements. Design and Planning come next, developing detailed designs and schedules. The Implementation phase involves developing and integrating the components. Testing and Validation ensure the project meets all requirements through rigorous testing. Deployment then moves the project to the production environment, followed by Maintenance and Evaluation to ensure ongoing performance and user satisfaction. This process is visualized in a flow chart, moving sequentially from one phase to the next, ensuring each stage is completed before proceeding.

3.2 Overall Flowchart

Figure 3.1 shows a flowchart that illustrates a structured approach to project planning, starting with defining the project objectives, followed by conducting a literature review to gather relevant information. Afterward, the project scopes are determined, with a crucial decision point to assess whether the scopes are well-defined. If they are satisfactory, the project proceeds; if not, the scopes are reassessed and refined. Once the scopes are approved, the project moves forward, concluding the planning phase and transitioning to

execution. This method ensures a thorough and well-organized planning process, enhancing the project's potential for success.

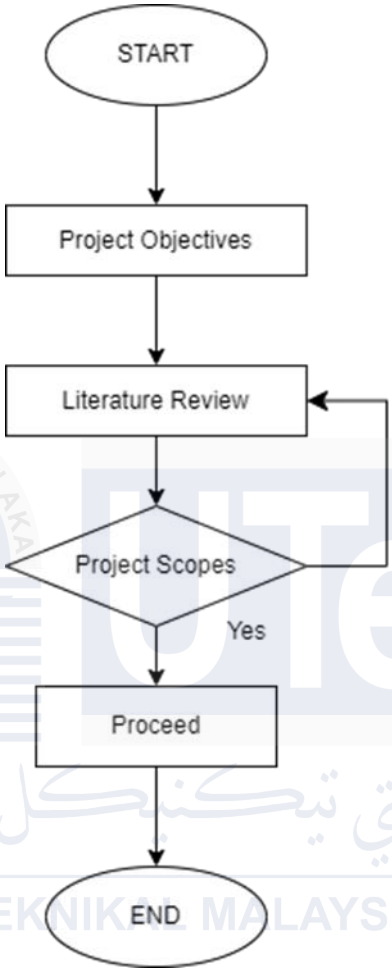


Figure 3.1 Overall Flowchart

3.3 System Flowchart

Figure 3.2 outlines a detailed process for disease detection using CNN. It begins with a dataset of images that are segregated into categories, such as healthy and diseased samples, and then pre-processed to standardize features like size and format, ensuring they are suitable for model training. The pre-processed data is used to train a CNN, which learns to identify patterns and features corresponding to different categories. Separately, a test image (not part of the training dataset) is provided as input, undergoing similar pre-processing to ensure compatibility with the trained model. The test image is converted into

a numerical array format and passed through the CNN for classification. If the CNN detects a disease, the system identifies the specific condition and displays appropriate remedies or recommendations. If no disease is detected, the image is classified as healthy. The process concludes after displaying the result.

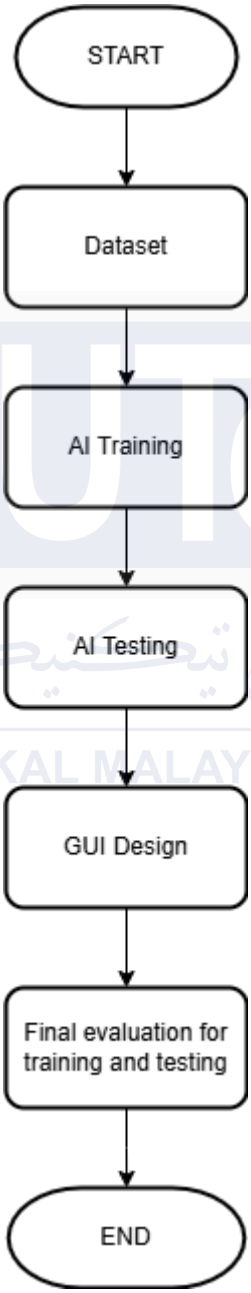


Figure 3.2 System Flowchart

3.4 Dataset

Web-sourced plant disease datasets, such as the PlantVillage dataset, provide high-quality images of plant leaves with annotations indicating whether they are healthy or diseased and specifying the disease type. These datasets, available on platforms like Kaggle or academic repositories, typically contain diverse samples across multiple plant species and diseases, organized into labeled folders. They are often used for training and evaluating machine learning models but may include limitations such as imbalanced classes or images captured under controlled conditions that do not fully reflect real-world scenarios. Before use, it is essential to ensure the dataset aligns with project goals and complies with licensing terms. Figure 3.3 shows a Healthy Grape Leaf, figure 3.4 shows a Grape Black Rot and figure 3.5 shows a Grape Leaf Blight.

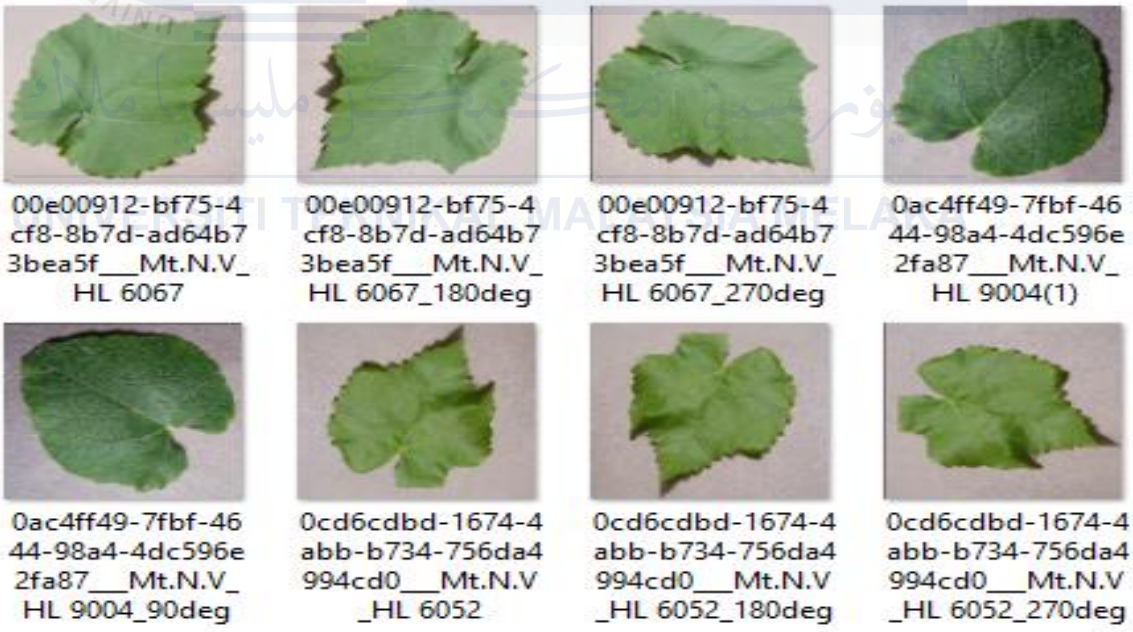


Figure 3.3 Healthy Grape Leaf

Figure 3.4 Grape Black Rot

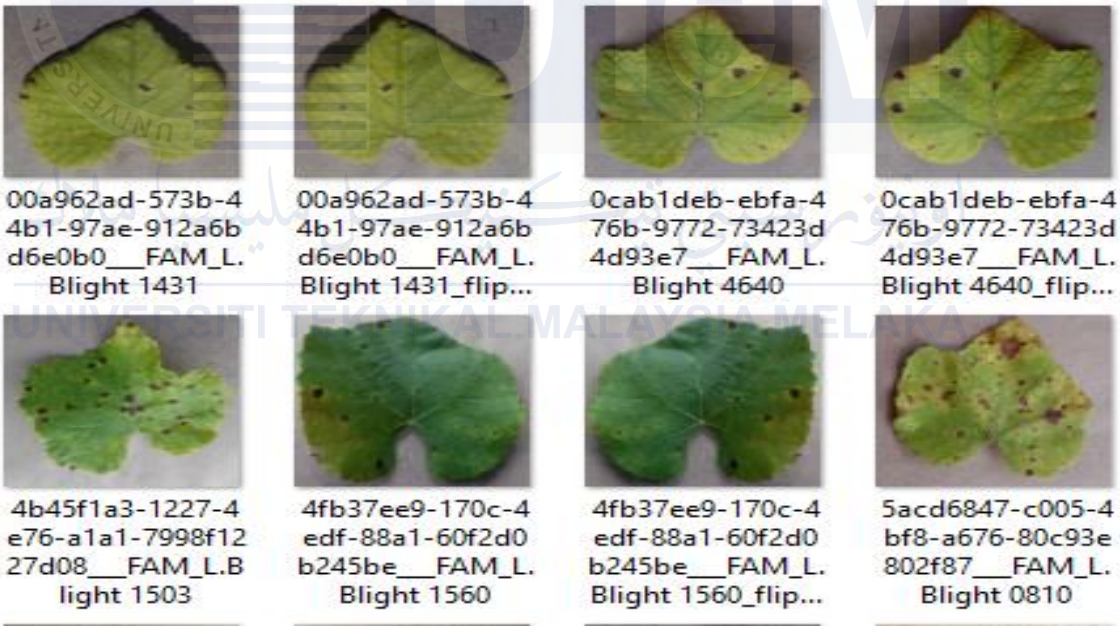


Figure 3.5 Grape Leaf Blight

3.5 AI Training

AlexNet is a convolutional neural network (CNN) introduced in 2012 by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, which revolutionized image recognition by winning the ImageNet competition with a significant reduction in error rates. It consists of 8 layers (5 convolutional and 3 fully connected) and introduced key innovations like ReLU

activation, dropout for regularization, overlapping max pooling, and GPU acceleration for faster training. AlexNet processes 227x227 color images and paved the way for modern deep learning models in computer vision.

Figure 3.6 shows algorithm training to set up the AlexNet training pipeline to initialize the necessary variables. This includes specifying the path to the dataset folder using a variable called `dataFolder`. Additionally, the input size for the images is defined to match the requirements of AlexNet, which expects images of size 227 x 227 pixels. These constants and paths ensure that the model and data preprocessing align correctly, forming the foundation for subsequent operations.

To prepare the data for training, the dataset is loaded into MATLAB using the `imageDatastore` function. This automatically assigns labels to images based on the folder names. The dataset is then divided into training and validation sets, with 70% allocated for training (`imdsTrain`) and 30% for validation (`imdsValidation`). Data augmentation is applied to the training set to introduce variability, such as random reflections and scaling, improving the model's robustness. Meanwhile, the validation set is resized to the required input dimensions without augmentation.

Before training, it’s helpful to visualize the training data to confirm that the data loading process was successful. A random selection of nine images from the training dataset is displayed in a grid. This optional but informative step ensures the images are correctly labeled and formatted, giving confidence in the correctness of the data pipeline before proceeding to model training.

AlexNet, a pre-trained deep learning model, is customized to fit the specific dataset. The base layers of AlexNet are retained, excluding the final three layers, which are tailored to its original dataset. These layers are replaced with a fully connected layer with neurons matching the number of dataset classes, a softmax layer for probability outputs, and a

classification layer for label prediction. This modification enables AlexNet to specialize in the new dataset while leveraging its pre-trained features.

Configuring training parameters is crucial for optimizing performance. The Stochastic Gradient Descent with Momentum (SGDM) optimizer is selected for efficient training. Key parameters include a mini-batch size of 32 images, a maximum of 10 epochs to limit the number of passes through the dataset, and an initial learning rate of 0.0001 to ensure steady convergence. Validation data is incorporated into the training process to monitor the model's performance and prevent overfitting.

Using the configured parameters and the modified AlexNet structure, the model is trained with the augmented training data. This step involves updating the model's weights to minimize the loss function, thereby improving its ability to classify images accurately. After training, the resulting model, named `netTransfer`, is saved as a `.mat` file (`PDC_Train.mat`) for future use, preserving the trained network and its parameters.

The model's performance is evaluated on the validation set to measure its accuracy. Validation images are classified using the trained model, and the predictions are compared against the actual labels. The resulting accuracy percentage quantifies the model's ability to generalize to unseen data. This step provides insights into the model's reliability and highlights areas where it may require further fine-tuning.

Finally, the model's predictions are visualized to assess its real-world application. Four random images from the validation set are selected and displayed along with their predicted class labels. This step demonstrates the model's ability to correctly classify images, offering a qualitative view of its effectiveness and helping identify specific cases where it may struggle. Figure 3.7 shows the architecture of AlexNet. Figure 3.8 displays the predictions of a machine learning model classifying grapevine leaf health. It includes four samples: the top two leaves are predicted as healthy, showing no visible disease symptoms,

while the bottom-left leaf is identified as having "Grape Black Rot," characterized by dark lesions, and the bottom-right leaf is predicted to have "blight," marked by browning and damage.

The setup illustrates the model's ability to distinguish between healthy and diseased leaves based on visible features such as colour and texture. Figure 3.9 illustrates the training progress of a machine learning model over 10 epochs, with accuracy and loss metrics tracked. The top graph shows the accuracy starting lower and quickly rising, stabilizing near 99%, while the bottom graph depicts the loss starting high and decreasing sharply, leveling off at a low value, indicating effective learning. The results panel highlights a validation accuracy of 99.30%, achieved in 18 minutes and 42 seconds on a single CPU with a constant learning rate of 0.001. The smooth curves and stabilized metrics suggest successful training and model convergence.

Algorithm: Training Process Using AlexNet

Input: Dataset directory (`dataFolder`)
Output: Trained AlexNet model (`PDC_Train.mat`)

1. **Initialize Variables**
   - Set `dataFolder` to the path of the dataset folder.
   - Define `inputSize` = `[227, 227]`.
2. **Load and Preprocess Data**
   - Load dataset:
     `imds ← Load images from dataFolder with labels based on folder names.`
   - Split data into training and validation sets:
     `[imdsTrain, imdsValidation] ← Split imds into 70% training and 30% validation.`
   - Create image augmenter:
     `imageAugmenter ← Configure augmenter with random reflection and scaling.`
   - Apply augmentation:
     - `augmentedTrain ← Resize and augment images from imdsTrain.`
     - `augmentedValidation ← Resize images from imdsValidation.`
3. **Visualize Training Data**
   - For `i` from 1 to 9:
     - Randomly select an image from `imdsTrain`.

- Display the image.

4. **Modify Pre-trained AlexNet**
   - Load pre-trained model:
     `net ← Load AlexNet.`
   - Extract base layers:
     `layersTransfer ← All layers except last three of net.`
   - Get number of classes:
     `numClasses ← Count unique labels in imdsTrain.`
   - Add new layers:
     - Fully connected layer with `numClasses` neurons.
     - Softmax layer.
     - Classification layer.
5. **Set Training Options**
   - Define training options:
     - Use stochastic gradient descent (SGDM).
     - Set MiniBatchSize = 32.
     - Set MaxEpochs = 10.
     - Set InitialLearnRate = 1e-4.
     - Include validation data.
6. **Train the Model**
   - Train network:
     `netTransfer ← Train with augmentedTrain, layers, and training options.`
   - Save trained model:
     `Save netTransfer as PDC_Train.mat.`
7. **Evaluate the Model**
   - Classify validation images:
     `[YPred, scores] ← Classify augmentedValidation using netTransfer.`
   - Compute accuracy:
     `accuracy ← Mean of correct predictions.`
   - Display accuracy.
8. **Display Predictions**
   - Randomly select 4 images from `imdsValidation`.
   - For each image:
     - Display the image.
     - Display the predicted label.

Figure 3.6 Algorithm of training process using AlexNet

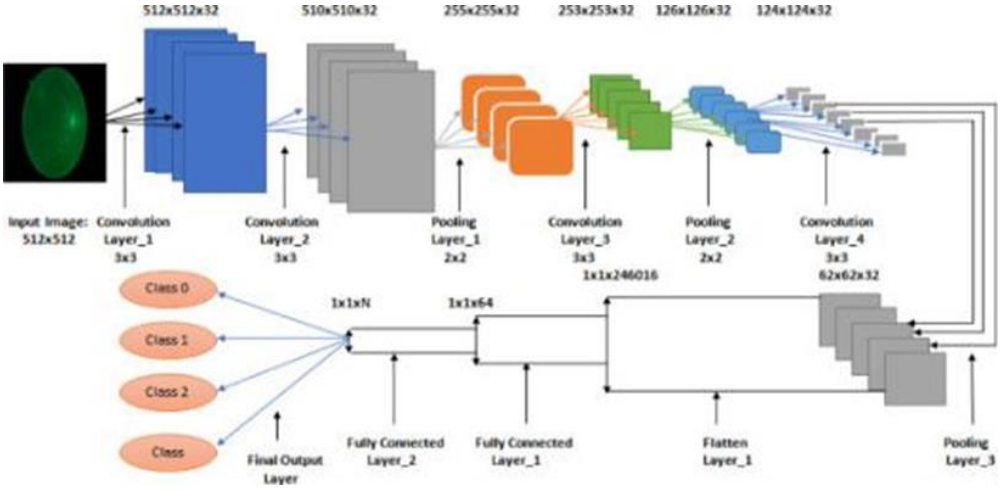


Figure 3.7 Architecture of AlexNet

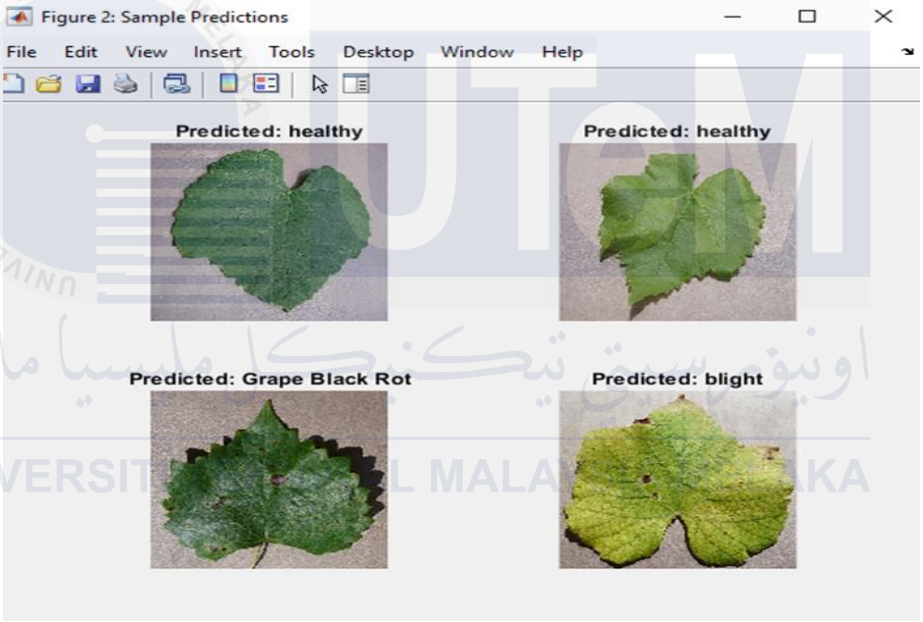


Figure 3.8 Sample Prediction

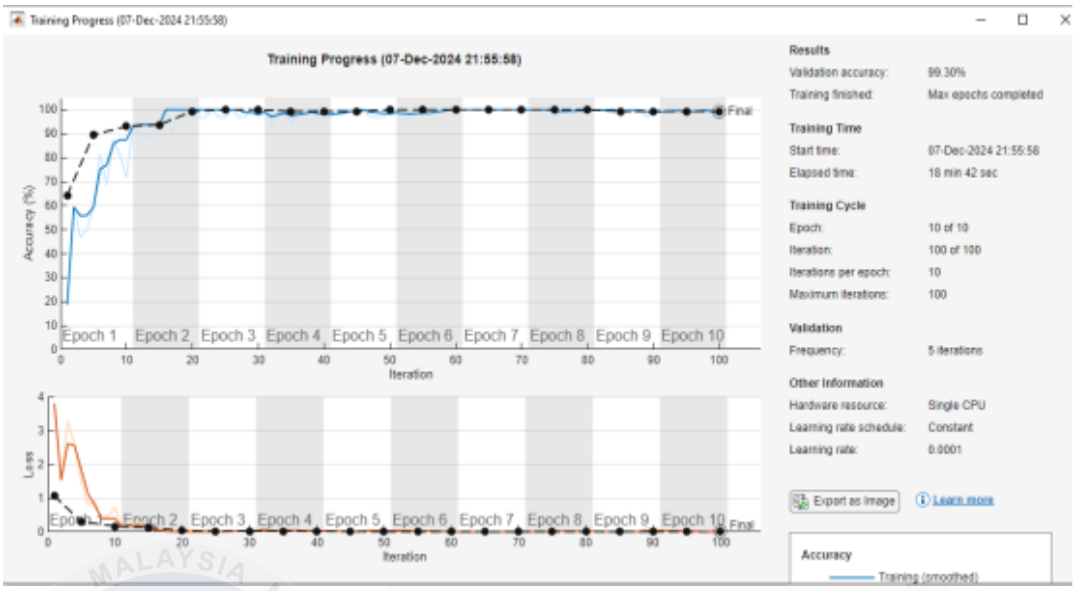


Figure 3.9 Training Progress

3.6 AI Testing

Figure 3.10 shows the algorithm for Plant Disease Detection Systems. The first step in the classification process is to load the trained AlexNet model from the PDC_Train.mat file. This file contains the pre-trained model, stored as the netTransfer variable. Upon loading the file, the system checks for the presence of the netTransfer variable to ensure the model has been correctly saved and is accessible. If the variable is missing, an error message is displayed to inform the user, and the process is terminated to prevent further execution without a valid model. This validation step is crucial to avoid runtime errors and ensure smooth operation during classification tasks.

Call back functions in the GUI are designed to handle user actions effectively, ensuring smooth operation and interaction. The "Browse Image" function allows the user to load a leaf image into the system. When triggered, it prompts the user to select an image file, which is then resized to [256, 256] for consistent display within the GUI. The total number of pixels in the image is stored for later calculations, such as determining the affected area.

The image is then displayed on the designated axis, axImage, providing a clear view for the user.

The "Detect Disease" function utilizes the trained AlexNet model to classify the loaded leaf image. Before proceeding it, verifies that an image has been loaded; if not, an error message is displayed. The image is resized to [227, 227], the input size required by AlexNet, and classified using the trained model (netTransfer). The predicted disease type and its confidence score are retrieved, and the GUI is updated accordingly. The classified image is displayed alongside the confidence score and disease type, ensuring the user has a clear understanding of the model's predictions.

The "Analyze Features" and "Display Cluster" functions focus on image segmentation and cluster analysis. In the "Analyze Features" step, the loaded image is converted to LAB color space, and the a and b channels are extracted for clustering. K-means clustering segments the image into three clusters, which are saved as separate segmented images. The "Display Cluster" function allows the user to select a cluster from the popup menu and view its details. Each cluster highlights specific regions: Cluster 1 often identifies diseased or rot-affected areas, Cluster 2 represents intermediate regions like early-stage disease or shadows, and Cluster 3 corresponds to healthy areas. If a cluster is selected, the system calculates and displays the affected area's percentage and shows the segmented cluster on axSegment. If no cluster is selected, the outputs are cleared to avoid confusion. Figure 3.11 shows the three-segmentation cluster.

The "Remedy" callback is designed to provide users with solutions for the detected disease. Before proceeding, the system verifies that an image has been loaded and disease detection has been successfully completed. If these conditions are not met, an error message is displayed to guide the user. Using the detected disease type (YPred), the system attempts to locate a corresponding remedy file that contains treatment or prevention guidelines. If the

remedy file is found, it is opened in the default text editor for easy access. However, if no matching file is found, an error message is displayed, informing the user that the remedy information is unavailable. This functionality ensures users receive relevant and actionable insights based on the classification results. Figure 3.12 shows the example of remedy.

Algorithm: Plant Disease Detection System

Input: Leaf image file, trained model (`PDC_Train.mat`), remedy files.
Output: Disease type, prediction confidence, affected area, remedy information.

1. **Initialize GUI**
   - Create a figure for the GUI with appropriate size, color, and layout.
   - Define axes for:
     - Leaf image display (`axImage`)
     - Affected area percentage (`axAffection`)
     - Prediction confidence (`axPrediction`)
     - Disease type (`axDisease`)
     - Segmented region of interest (ROI) (`axSegment`)
   - Add popup menu for cluster selection.
   - Add buttons for "Browse," "Detect Disease," "Analyze Features," and "Remedy."
2. **Load Trained Model**
   - Load `PDC_Train.mat`.
   - Check for the variable `netTransfer`.
   - If not found, display an error and exit.
3. **Callback Functions**

   (a) Browse Image

   - Prompt the user to select an image file.
   - Load and resize the image to `[256, 256]`.
   - Store the total number of pixels in the image.
   - Display the loaded image on `axImage`.

   (b) Detect Disease

   - If no image is loaded, display an error and exit.
   - Resize the image to the input size of the model (`[227, 227]`).
   - Classify the image using `netTransfer`.
   - Extract the predicted label (`YPred`) and confidence score.
   - Update the GUI to display:
     - The detected image on `axImage`.
     - The prediction confidence on `axPrediction`.

- The predicted disease type on `axDisease`.

(c) Analyze Features

- If no image is loaded, display an error and exit.
- Convert the image to LAB color space.
- Extract the `a` and `b` channels and reshape for clustering.
- Perform k-means clustering with 3 clusters.
- Generate segmented images for each cluster.

(d) Display Cluster

- Get the selected cluster from the popup menu.
- If a cluster is selected:
  - Calculate the affected area as the percentage of pixels in the selected cluster.
  - Display the segmented cluster on `axSegment`.
  - Display the affected area percentage on `axAffection`.
- If no cluster is selected, clear `axSegment` and `axAffection`.

(e) Remedy Callback

- If no image is loaded or disease detection is not performed, display an error and exit.
- Get the detected disease type from `YPred`.
- Match the disease type to a corresponding remedy file.
- If a remedy file exists:
  - Open the remedy file in the default text editor.
- If no remedy file is found, display an error message.

4. **End Program**

Figure 3.10 Algorithm of Plant Disease Detection System.

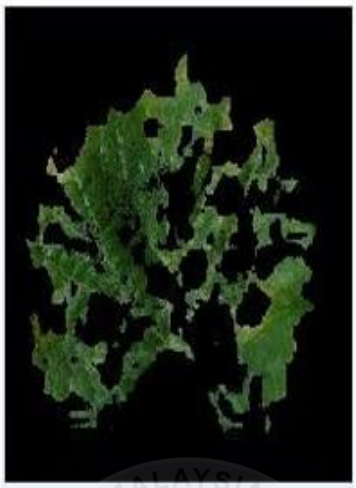
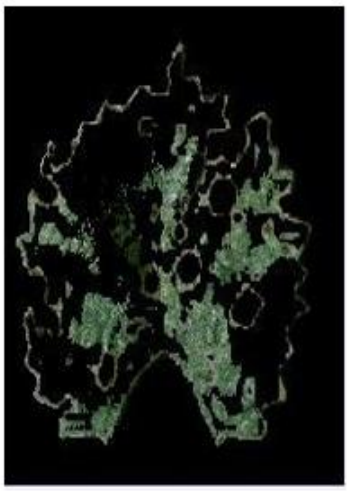
Cluster1	Cluster2	Cluster3
		
Cluster 1: often identifies diseased or rot-affected areas of Grape Black Rot	Cluster 2: represents intermediate regions like early-stage disease or shadows of Grape Black Rot.	Cluster 3 corresponds to healthy areas. If a cluster is selected of Grape Black Rot.

Figure 3.11 Segmentation Cluster

BLACK ROT REMEDY - Notepad

File Edit Format View Help

BLACK ROT REMEDY

Once your cruciferous vegetables are growing, be sure to fertilize them appropriately. In particular, inadequate nitrogen can predispose plants to black rot. Also, be gentle with cruciferous vegetables to prevent any wounds that might serve as entry points for Xcc. DO NOT use a sprinkler to water your vegetables as this can splash Xcc from plant to plant. Instead, use a soaker or drip hose that applies water directly to the soil. Avoid working with plants when they are wet to help limit spread of Xcc. If severe black rot develops, promptly remove symptomatic plants as well as all cruciferous plants within a three to five foot radius. Dispose of these plants by burning (where allowed by local ordinance), burying or composting them. If you decide to compost, make sure your compost pile heats to a high enough temperature and that any infested material decomposes for at least one year before it is reincorporated into your garden.
Finally, decontaminate any pots, tools, or other gardening items that have come into contact with Xcc-infected plants or Xcc-infested debris by treating them for at least 30 seconds with 70% alcohol (preferable for metal tools because of its less corrosive properties) or 10% bleach. Rubbing alcohol and many spray disinfectants typically contain approximately 70% alcohol. If you use bleach on metal tools, be sure to thoroughly rinse and oil them after use to prevent rusting.
REFERENCE
https://hort.extension.wisc.edu/articles/black-rot-crucifers/

Figure 3.12 Example of Remedy

3.7 GUI Design

Figure 3.13 shows a MATLAB App Designer interface where a UI Figure and an Axes component are configured. The Axes is titled "Leaf Image," with X, Y, and Z axis labels and Helvetica font styled as bold with a size of 12. The UI Figure has a background color set to [0.94, 0.94, 0.94], a position defined as [100, 100, 640, 480], and settings for resizing and auto-resizing children enabled. The interface is designed to display plots or images, with the axes centrally aligned for clear visualization.

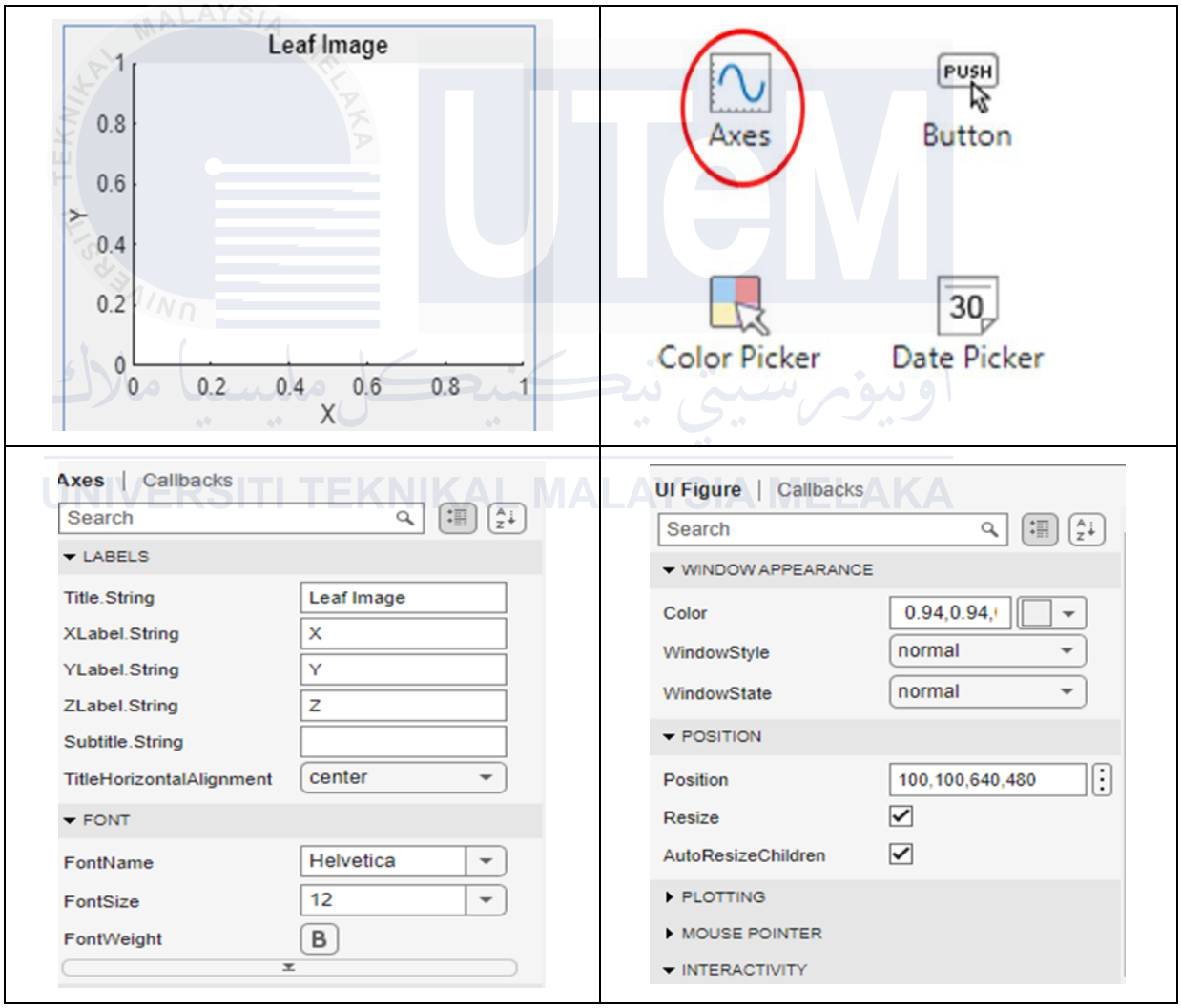


Figure 3.13 Axes In App Designer

This figure 3.14 showcases the Drop-Down component in MATLAB App Designer, configured to display a labeled menu with the text "Select Cluster" and options such as

"cluster1," "cluster2," and "cluster3," with "cluster1" currently selected. The drop-down menu is styled with Helvetica font, bold, size 12, a light-gray background ([0.96, 0.96, 0.96]), and black text. Positioned at [750, 550, 300, 30] within the UI figure, it fits well into the layout. The UI figure itself has a light-gray background ([0.94, 0.94, 0.94]), with resizing enabled to allow dynamic adjustments.

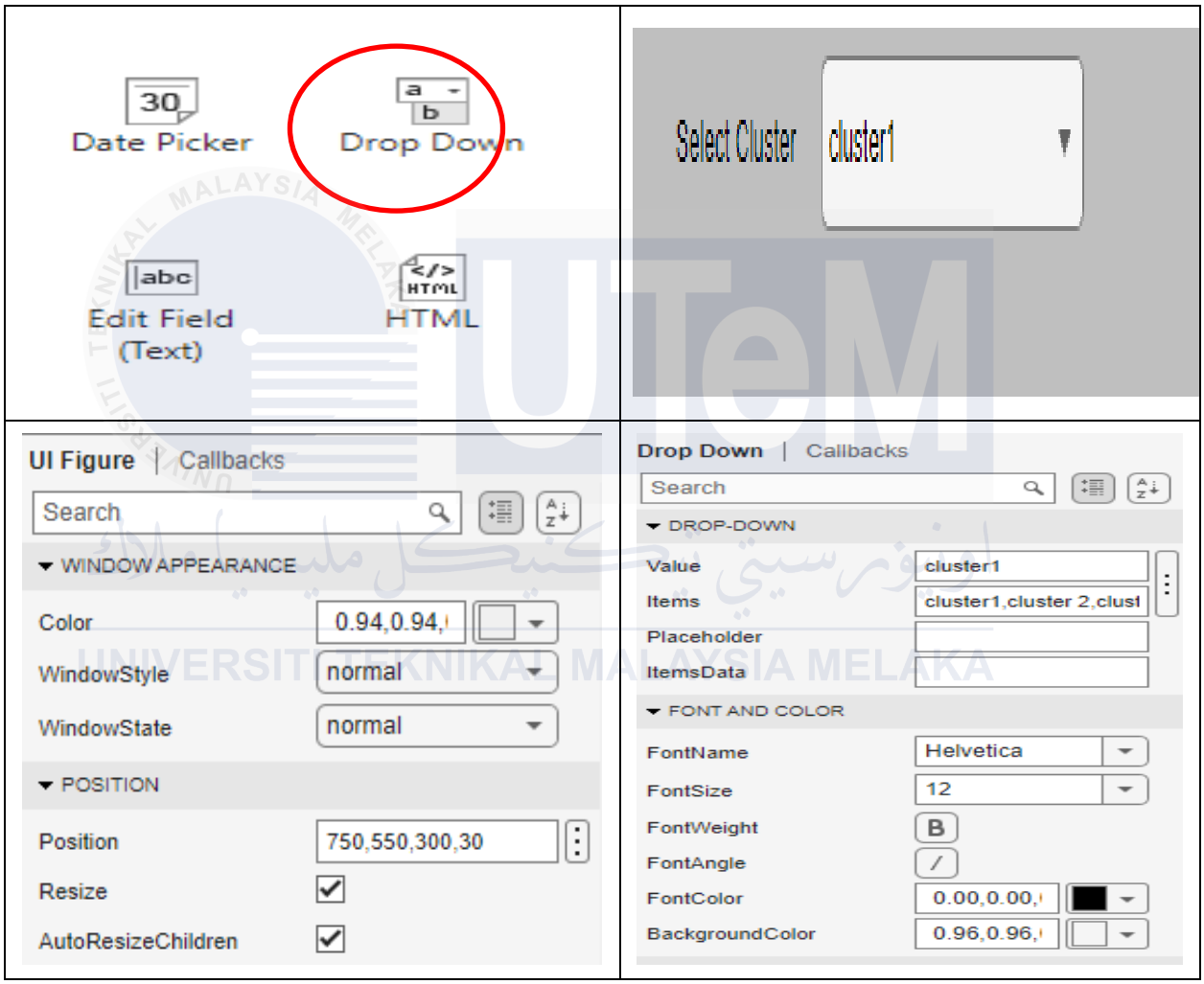


Figure 3.14 Drop Down Component in App Designer

Figure 3.15 shows representations of a button in MATLAB's App Designer: the left side depicts the button in runtime, where it is labeled "PUSH" and interacts with the user, while the right side shows the button in design mode, with a blue background, editable text ("Button"), and resizing handles for customization. There also a button in Plant Disease Detection GUI with the functions.

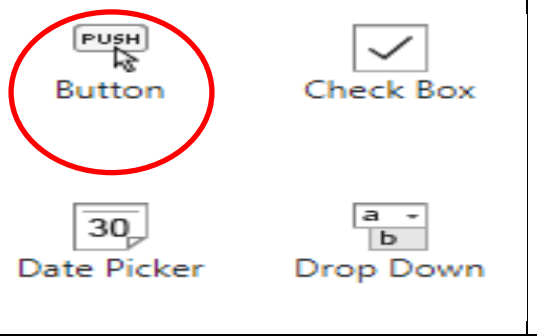
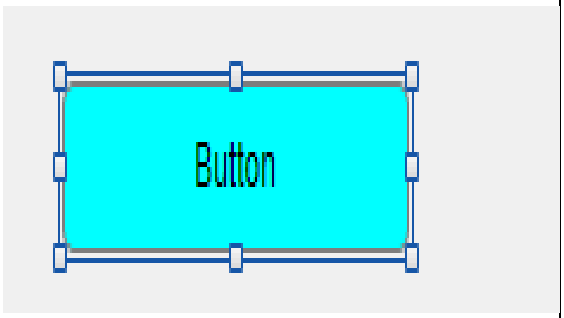
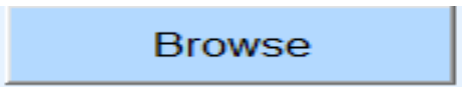
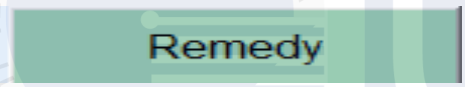
	
	"Browse" button for loading a leaf image into the system.
	"Detect Disease" button to classify the disease based on the input image.
	"Analyze Features" button to perform k-means clustering on the image for detailed analysis.
	"Remedy" button to display recommended solutions for the detected disease.

Figure 3.15 GUI Push Button

For label ‘Prediction Confidence (%)’, Affected Area (%), ‘Disease Type’ and the ‘Plant Disease Detection GUI’ tittle, I use the label function in App Designer in MATLAB app. Figure 3.16 shows the label function in App Designer.

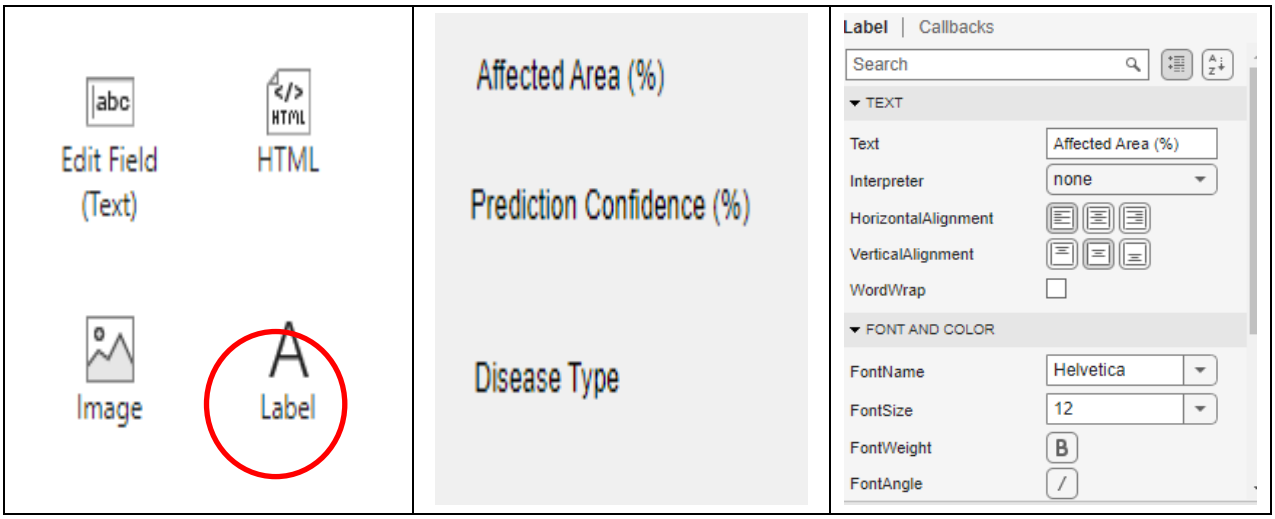


Figure 3.16 Label function

Figure 3.17 shows result for GUI design created in MATLAB's App Designer. It includes a title at the top ("Plant Disease Detection GUI") and features components such as

axes for displaying the leaf image and selected clusters (ROI), labels for displaying prediction confidence, affected area, and disease type, and a dropdown menu for selecting clusters. At the bottom, buttons like "Browse," "Detect Disease," "Analyze Features," and "Remedy" enable users to upload images, detect diseases, analyze features, and view remedies, all within a clean, light blue interface.

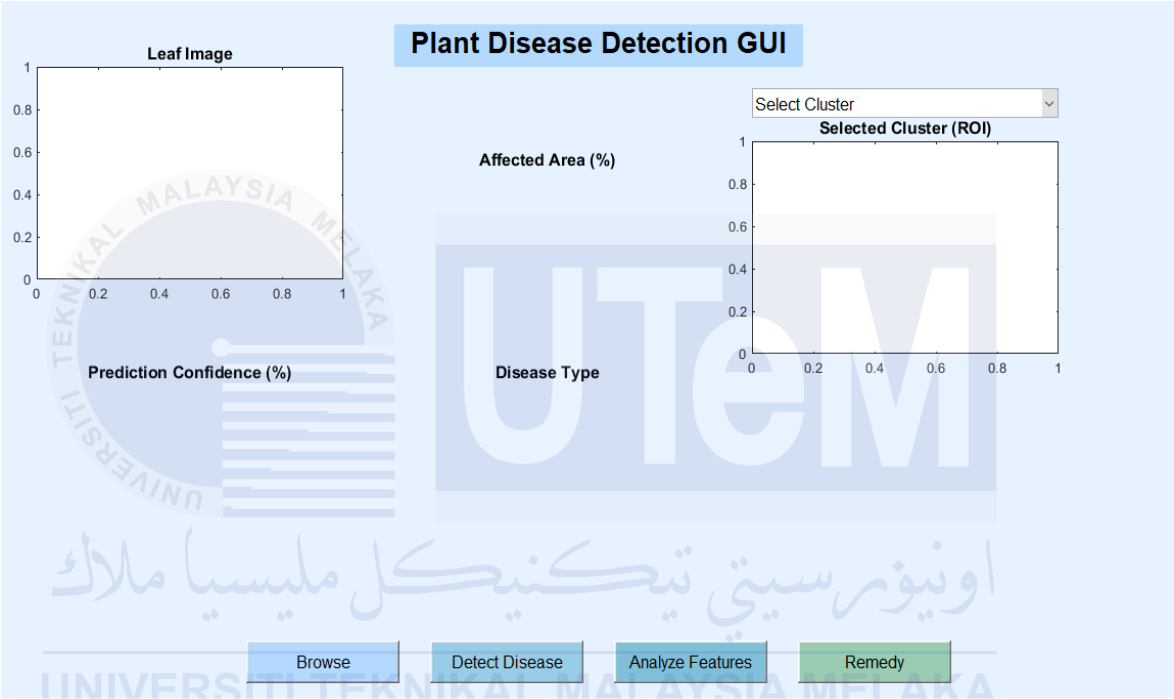


Figure 3.17 GUI Design Result

3.8 Final Evaluation for Training and Testing

Figure 3.18 outlines a systematic process, beginning with the **START** node to initiate the workflow. The next step involves obtaining a result from a **Graphical User Interface (GUI)**, which likely represents some computational or automated output. Once the GUI provides a result, the process moves to the **Verification of Result and Manual Calculation** stage, where the obtained result is thoroughly checked and cross-verified using manual calculations to ensure accuracy and reliability. This verification step is crucial for validating the correctness of the automated output. Finally, the process concludes at the **END** node, signifying the completion of all required tasks in the workflow.

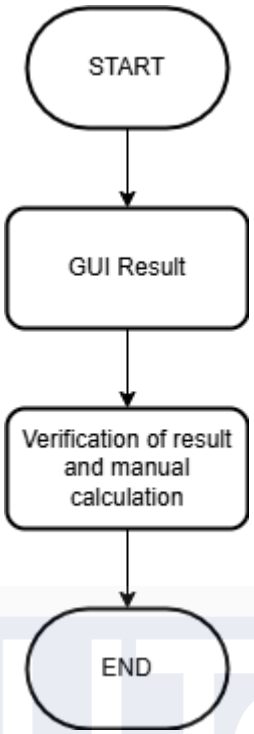


Figure 3.18 Final Evaluation flowchart

3.8.1 GUI Result

Figure 3.19 represents a user interface for a **Browse Button**. At the top, there is a **"Browse"** button, which allows users to upload or select images. Below it, the interface displays three loaded images of leaves, each labelled **"Loaded Image".** These images appear to showcase different conditions of the leaves, possibly for purposes like analysis, comparison, or classification (e.g., identifying healthy versus diseased leaves). The layout suggests that the system is designed to handle multiple images and display them in an organized format for further interaction or processing.

Figure 3.20 shows the result of a **"Detect Disease" button**, where the system analyzes an input (e.g., a leaf) and provides its health status. The output includes the “Prediction Confidence (%)”, which is “100.00%”, indicating complete confidence in the result, and the “Disease Type”, which is identified as "healthy", meaning no disease is detected. The interface clearly presents the information in a simple and user-friendly format.

Figure 3.21 shows the function of **Analyze Feature button** illustrates the analysis of a leaf's affected areas, segmented into three clusters (regions of interest). Each cluster displays an image of the selected region and the corresponding percentage of the area affected, likely by damage or disease.

The results show that “Cluster 1” has 49.23% affected, “Cluster 2” has 18.27%, and “Cluster 3” has 32.43%, highlighting the varying severity of impact across the leaf. Figure 3.22 shows the function of **Remedy button** that are serves as a tool or feature that provides users with detailed guidelines or instructions for addressing specific plant diseases. In this context, clicking the **Remedy button** gives access to tailored solutions, such as the **Black Rot Remedy**, which focuses on proper fertilization, watering techniques, and sanitation, and the **Blight Leaf Remedy**, which emphasizes spacing, pest control, and chemical treatments. The button essentially acts as a quick reference for disease management strategies.

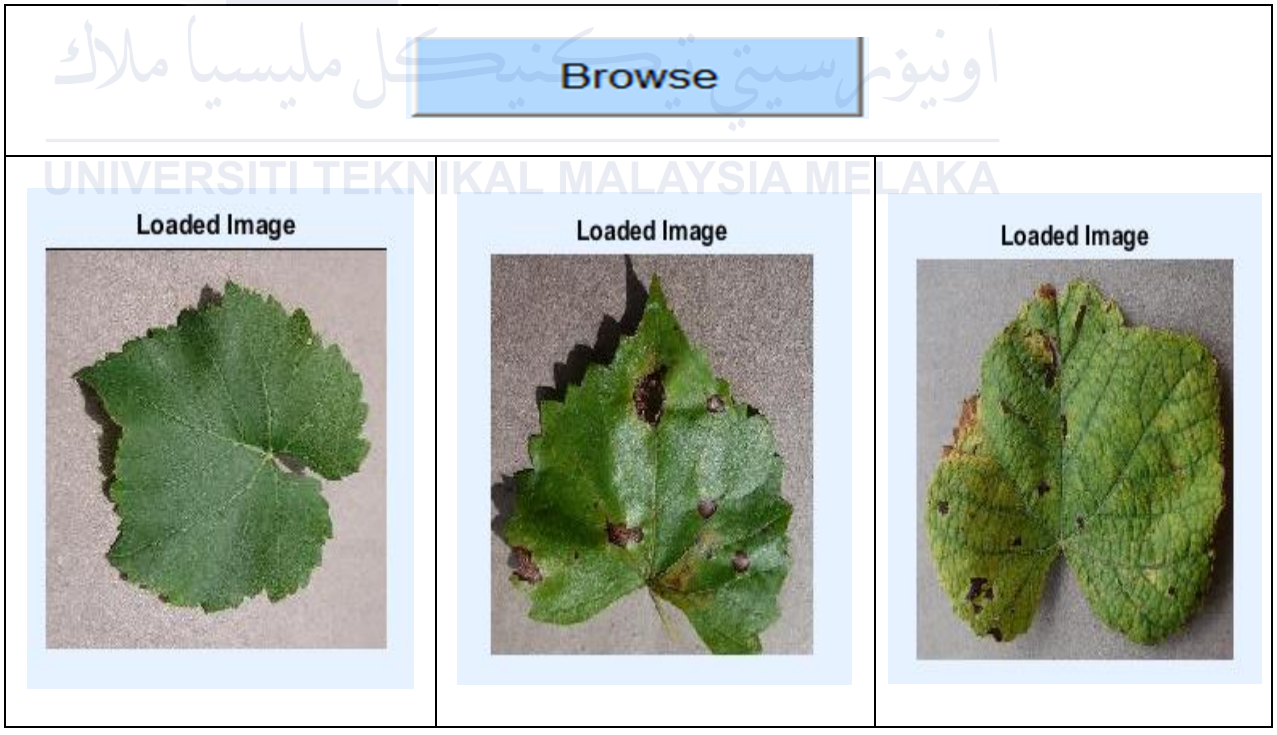


Figure 3.19 Browse Button Result

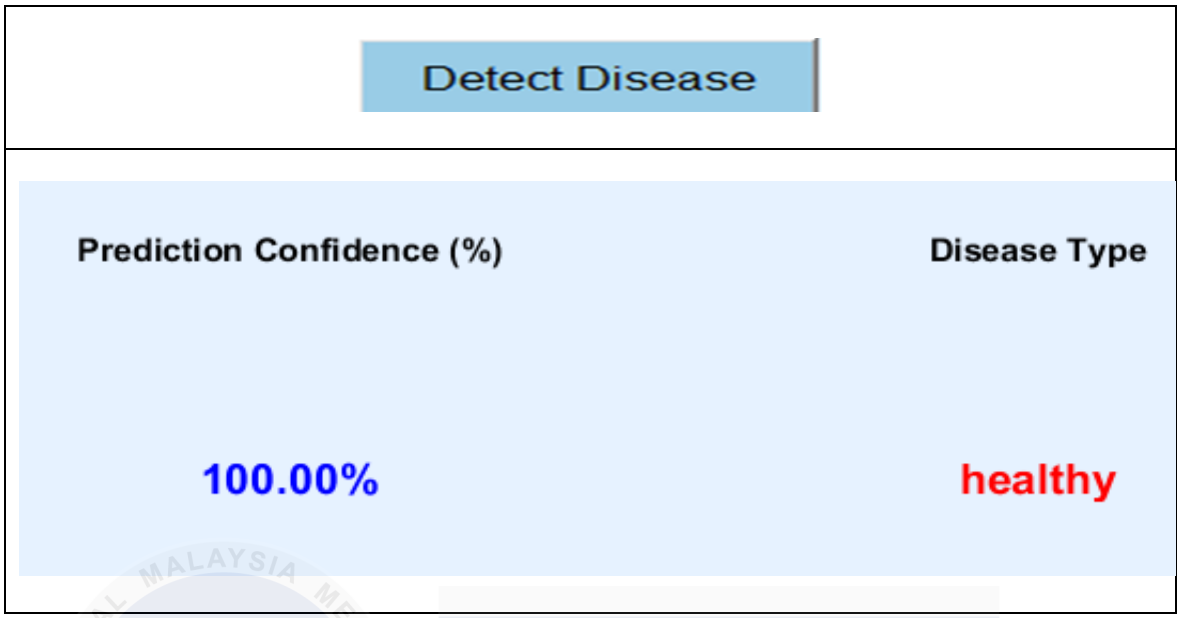


Figure 3.20 Detect Disease Button Result

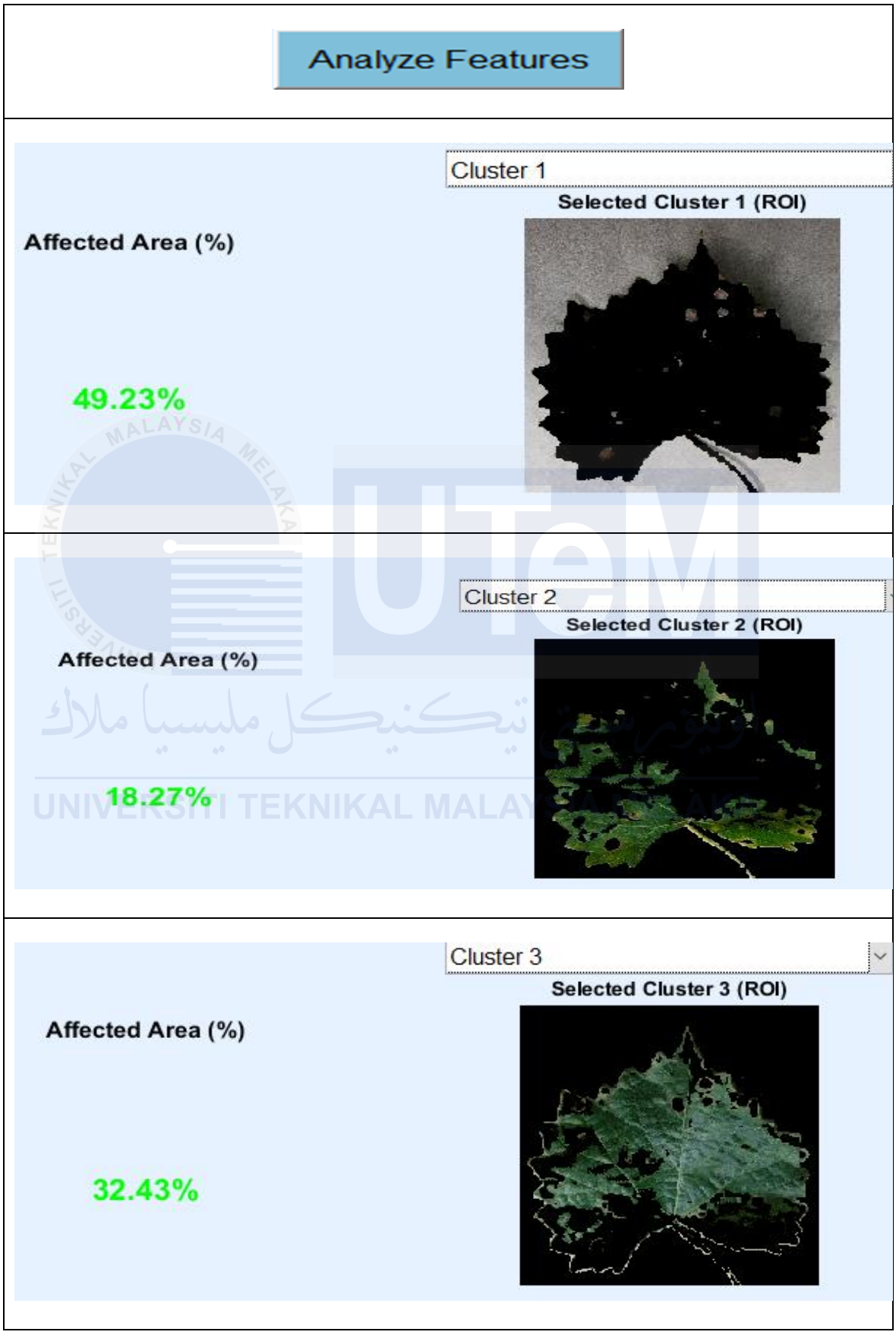


Figure 3.21 Analyze Features Button Result

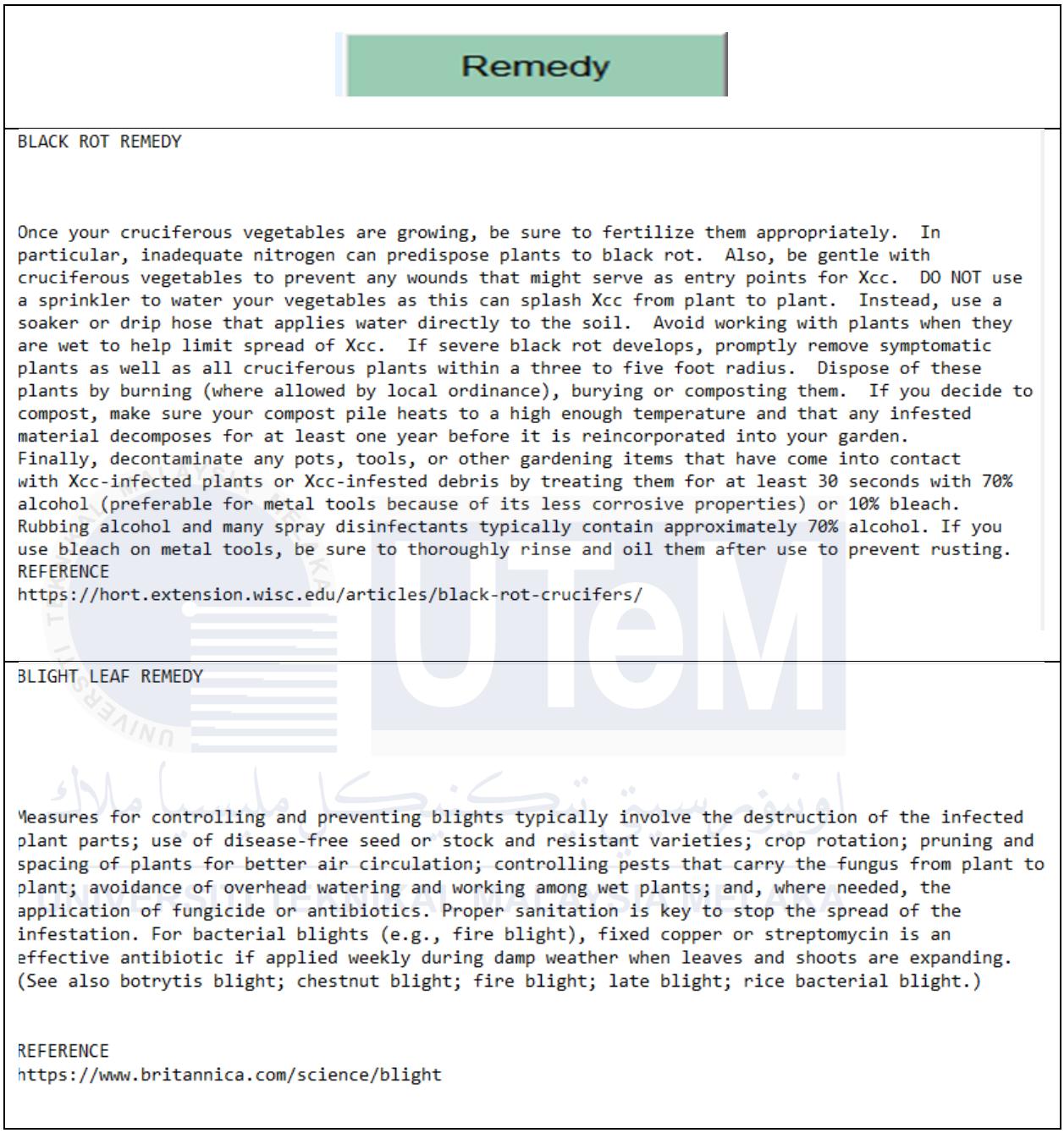


Figure 3.22 Remedy Button Result

3.8.2 Verification of Results and manual calculation

After the GUI displays the results, it is essential to verify their accuracy to ensure the system performs as expected. If the GUI displays 'healthy' for an image, the result should be cross-checked against the initial labelling of the test dataset. For instance, if the dataset contains 20 images of healthy plants, and the GUI consistently classifies all of them as

'healthy,' the results can be considered accurate. Maintaining a log of these outcomes is crucial for tracking the model's performance over time. This process should be repeated with datasets from various plants and different diseases to evaluate the system's adaptability and robustness across a wider range of scenarios.

The primary objective of the data analysis in this project is to determine the accuracy percentage of the plant disease detection system. This involves comparing the system's predictions with the actual conditions of the plants whether they are healthy or diseased and calculating the proportion of correct predictions. The process begins with **ground truth labelling**, where the dataset is accurately labelled, such as marking the 20 test images as 'healthy' if they represent healthy plants. Next, the **model predictions** are generated by using the trained detection model to classify each image as either 'healthy' or 'diseased.'

To evaluate the model's performance, the predictions are compared against the ground truth labels, and the number of correct matches is counted. The **accuracy percentage** is then calculated using the formula:

$$Accuracy = \left(\frac{Number\ of\ Actual\ Number - Number\ of\ Error}{Number\ of\ Actual} \right) x\ 100\% \qquad (3.1)$$

3.9 Summary

The methodology for plant disease detection using CNNs involves several key steps. Firstly, a diverse dataset of plant images, including healthy specimens and various diseased conditions, is collected. These images undergo preprocessing to standardize their size, color, and format. Subsequently, an appropriate CNN architecture is selected, and the model is trained on the preprocessed dataset, often employing techniques like transfer learning or fine-tuning. The trained model is then evaluated using validation datasets to assess its performance metrics. Further optimization of the model may be conducted through fine-tuning of hyperparameters and architecture. In the operational flowchart, input plant images

are preprocessed, and relevant features are extracted using the trained CNN model. These features are then classified into disease categories or healthy status, and the final output indicates the detected disease or plant health. Optional feedback loops can be incorporated to continuously improve the model's performance based on user feedback. This methodology ensures accurate and efficient plant disease detection, facilitating effective agricultural management practices.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

This section presents the outcomes of our study on plant disease detection using CNNs implemented in MATLAB. We begin by showcasing the results of our model, including key performance metrics such as predicted confidence and percentage of affected area. These metrics provide a comprehensive understanding of the model's effectiveness in identifying and classifying plant diseases from leaf images. Through detailed data analysis, we evaluate the CNN's performance across different types of diseases and compare its accuracy with other traditional methods. We also explore the impact of various preprocessing techniques and model hyperparameters on the results. This analysis helps identify the strengths and limitations of our approach.

In the discussion section, we interpret these findings in the context of practical agricultural applications. We discuss the potential benefits of implementing such a system for real-time disease monitoring and management, highlighting its advantages in terms of speed, accuracy, and scalability. Additionally, we address the challenges encountered during the study, such as handling imbalanced datasets and the need for high-quality annotated images. Finally, we propose future research directions to further enhance the model's robustness and applicability, including the integration of additional data types and the use of transfer learning to improve performance on smaller datasets.

4.2 Results and Analysis

Figure 4.1 shows a plant disease detection software analyzing the same leaf images, determined to be “healthy” with 0.00% affected area and 100.00% prediction confidence. The software confirms no disease presence in either leaf, as indicated by the “healthy” label in red text. The left panels display the analyzed images and affect area results, while the right panels explicitly state “NO REMEDY” since no treatment is needed for healthy leaves. Buttons for further analysis, such as "Browse," "Detect Disease," "Analyze Features," and "Remedy," are still visible but unused in this case.

Figure 4.2 displays a plant disease detection software (left) and a corresponding remedy guide in Notepad (right). In both cases, the software analyzes leaf images, identifying Grape Black Rot with 100% prediction confidence. The first leaf shows 49.22% affected area, while the second shows 32.42% and the third shows 18.29%. The software includes buttons for browsing, detecting diseases, analyzing features, and providing remedies. The remedy guide emphasizes proper fertilization to avoid nitrogen deficiency, preventing wounds on plants, and using a soaker or drip hose instead of sprinklers to reduce water splashes that spread the disease. It also advises working with plants when they are dry to help limit infection.

Figure 4.3 shows a graphical user interface (GUI) for plant disease detection and its corresponding remedy information. On the left side under "GUI RESULT," two examples of plant leaf analysis are displayed. Each analysis identifies the leaf’s disease as "blight" with 100% confidence, along with the affected area percentages (48.54%, 36.73% and 14.73%). The images also show clustering visualizations to highlight the affected regions. On the right side under "REMEDY," a Notepad window provides a textual remedy for blight, recommending measures such as the destruction of infected plant parts, crop rotation, proper spacing, avoiding overhead watering, and the use of fungicides or antibiotics. The remedy

emphasizes sanitation and prevention to control the spread of bacterial blight. This combined system helps users detect and analyze plant diseases and provides actionable solutions.Insert your content here.

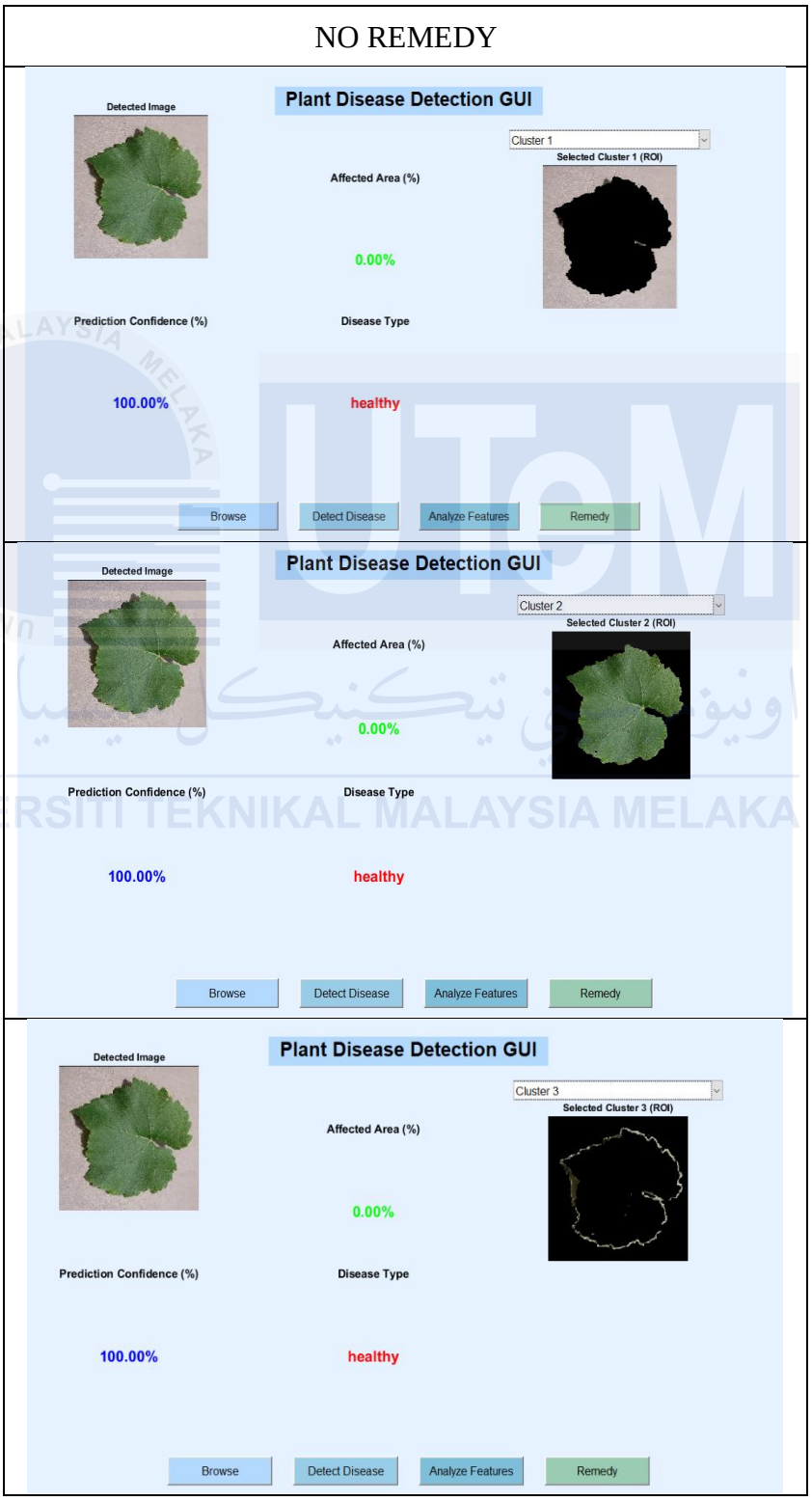


Figure 4.1 Result for Healthy Plant

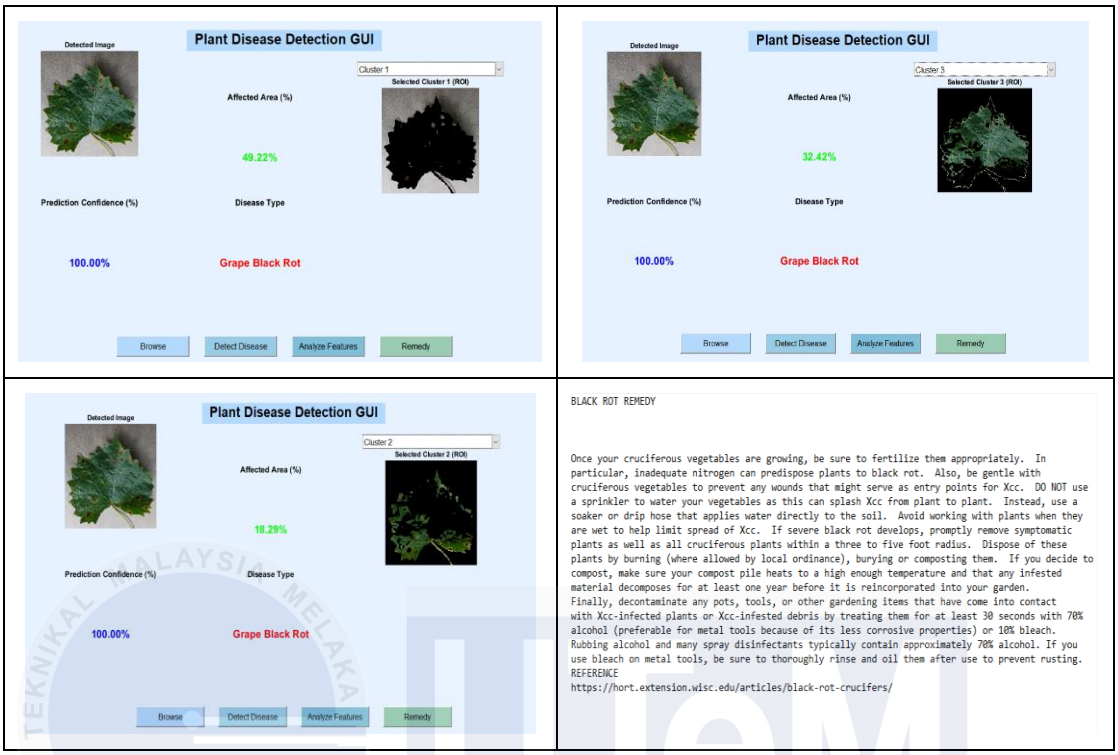


Figure 4.2 Result for Grape Black Rot

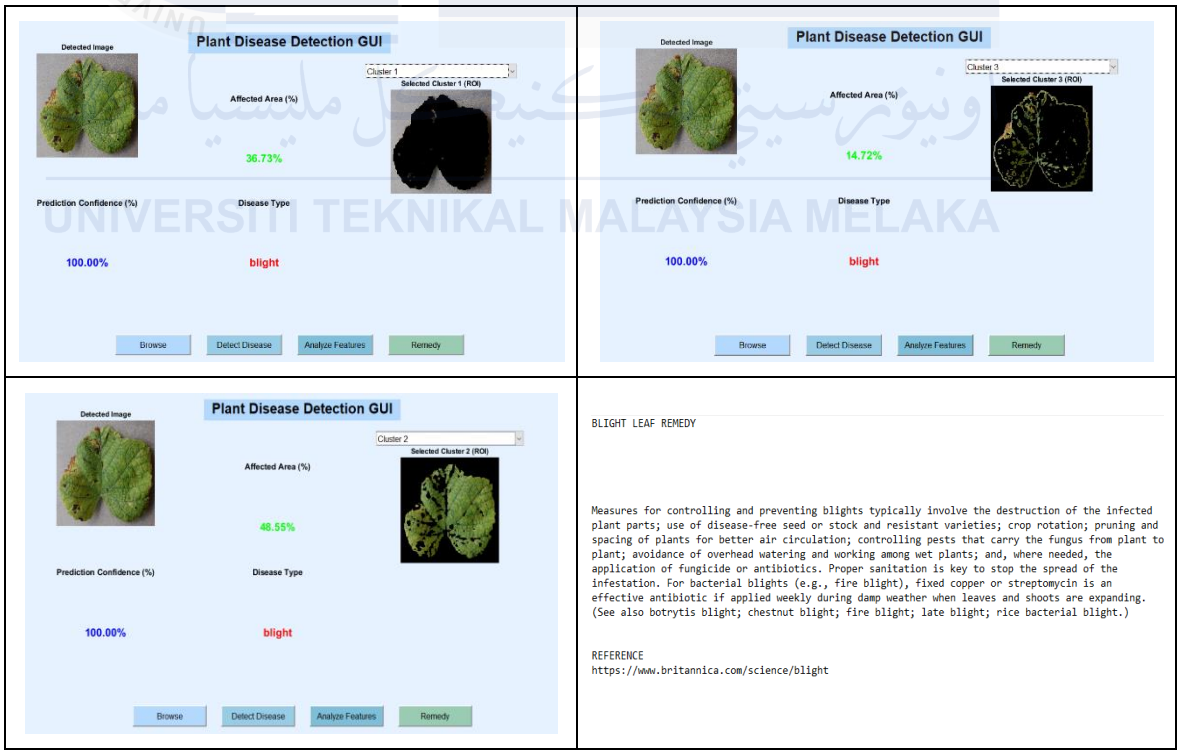


Figure 4.3 Result for Blight Leaf

Figure 4.4, 4.5 and 4.6 shows the test dataset for testing part. Each figure shows 20 dataset that had rename T1 to T60. Table 4.1 summarizes the results of a plant disease

detection experiment across 20 test samples. Samples T1 to T6 are healthy, with no affected areas and 100% detection confidence. Samples T7 to T20 show varying percentages of disease spread across three clusters (indicating different disease types or severities). The detection was successful for all samples except T20, which had a lower confidence of 96.42%.

Overall, the detection algorithm performed well, consistently achieving high confidence levels (mostly 99.85–100%). Table 4.2 presents the results of plant disease detection for test samples T21 to T40. Samples T21 to T31 are healthy, with no affected areas (all percentages are 0.00%) and detection confidence of 100%. Samples T32 to T40 show varying degrees of disease spread across three clusters, representing different disease types or severities (e.g., "Black Rot" or "Leaf Blight"). The detection was successful for all samples, with confidence levels ranging from 96.42% to 100%. The results demonstrate consistent and accurate detection for both healthy and diseased samples, with high confidence in the predictions.

Table 4.3 shows plant disease detection results for samples T41 to T60. Samples T41 and T42 are healthy, with no affected areas (all percentages are 0.00%) and 100% detection confidence. Samples T43 to T60 show varying degrees of disease spread across three clusters (indicating different disease types or severities like "Black Rot" or "Leaf Blight"). The detection was successful for all samples, with confidence levels ranging from 96.42% to 100%. The percentages highlight the distribution of the affected area among clusters, such as T43 with 55.82% in Cluster 3 and T44 with 60.53% in Cluster 1. The results demonstrate accurate detection for all samples, maintaining high prediction confidence.

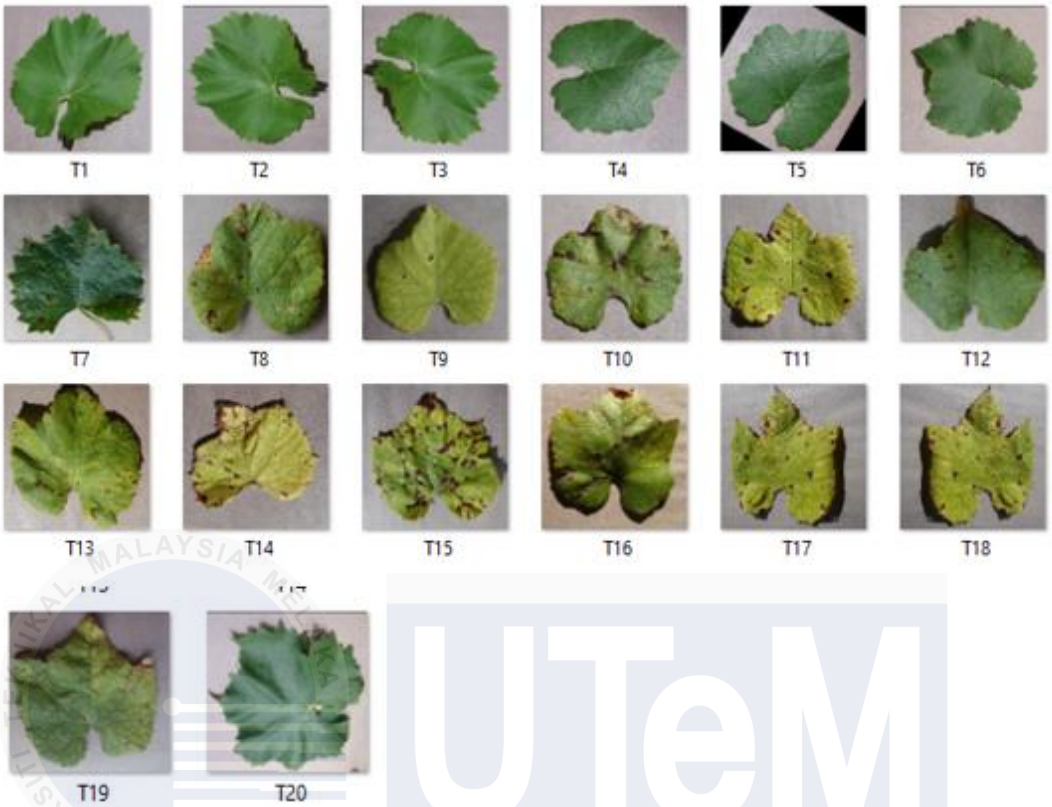


Figure 4.4 Test Dataset part 1

Table 4.1 Result of Plant Disease Detection part 1

Plant Disease Detection Result								
Test Data	Classify			Percentage of Affected Area (%)			Detection Successful	Predicted Confidence (%)
	Healthy	Black Rot	Leaf Blight	Cluster 1	Cluster 2	Cluster 3		
T1				0	0	0	Yes	100
T2				0	0	0	Yes	100
T3				0	0	0	Yes	100
T4				0	0	0	Yes	100
T5				0	0	0	Yes	100
T6				0	0	0	Yes	100
T7				18.29	49.22	32.42	Yes	100
T8				48.55	36.73	14.72	Yes	100
T9				47.79	46.69	5.52	Yes	100
T10				37.59	46.09	16.33	Yes	100
T11				49.15	39.60	11.25	Yes	100
T12				32.95	22.03	45.03	Yes	99.94
T13				41.04	41.86	17.10	Yes	100
T14				56.26	11.11	32.63	Yes	99.97

T15				36.06	51.56	12.38	Yes	99.99
T16				22.17	51.16	26.67	Yes	99.85
T17				35.44	55.05	9.52	Yes	100
T18				55.17	34.95	9.89	Yes	100
T19				44.89	24.30	30.81	Yes	100
T20				36.91	40.43	22.65	No	96.42

Figure 4.5 Test Dataset part 2

Table 4.2 Result of Plant Disease Detection part 2

Plant Disease Detection Result								
Sample Test Data	Classify			Percentage of Affected Area (%)			Detection Successful	Predicted Confidence (%)
	Healthy	Black Rot	Leaf Blight	Cluster 1	Cluster 2	Cluster 3		
T21				0.00	0.00	0.00	Yes	100.00
T22				0.00	0.00	0.00	Yes	100.00
T23				0.00	0.00	0.00	Yes	100.00
T24				0.00	0.00	0.00	Yes	100.00
T25				0.00	0.00	0.00	Yes	100.00

T26				0.00	0.00	0.00	Yes	100.00
T27				0.00	0.00	0.00	Yes	100.00
T28				0.00	0.00	0.00	Yes	100.00
T29				0.00	0.00	0.00	Yes	100.00
T30				0.00	0.00	0.00	Yes	100.00
T31				0.00	0.00	00.00	Yes	100.00
T32				24.87	52.74	22.38	Yes	99.94
T33				66.61	16.86	16.53	Yes	100.00
T34				16.29	66.62	16.59	Yes	99.97
T35				53.46	19.74	26.27	Yes	99.99
T36				29.25	50.07	20.64	Yes	99.85
T37				26.60	51.86	21.54	Yes	100.00
T38				49.20	32.33	18.41	Yes	100.00
T39				18.48	49.21	32.26	Yes	100.00
T40				62.38	15.86	21.76	Yes	96.42

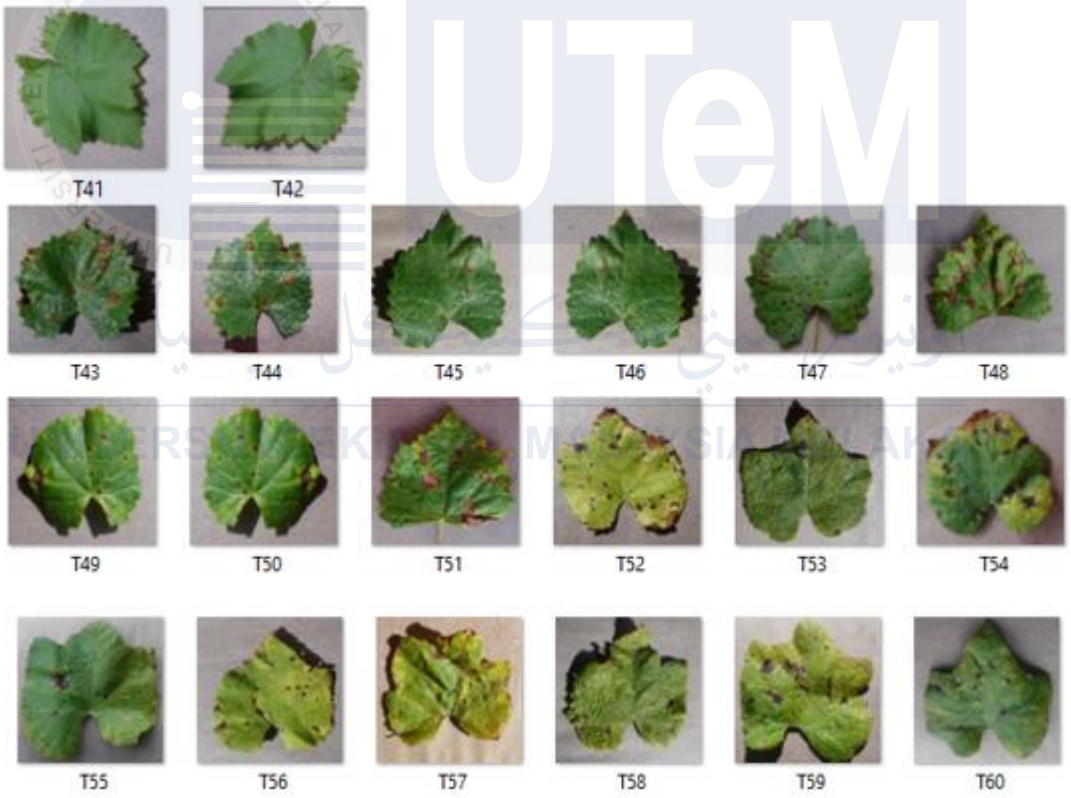


Figure 4.6 Test Dataset part 3

Table 4.3 Result of Plant Disease Detection part 3

Plant Disease Detection Result								
Sample Test Data	Classify			Percentage of Affected Area (%)			Detection Successful	Predicted Confidence (%)
	Healthy	Black Rot	Leaf Blight	Cluster 1	Cluster 2	Cluster 3		
T41				0.00	0.00	0.00	Yes	100.00
T42				0.00	0.00	0.00	Yes	100.00
T43				17.84	26.31	55.82	Yes	100.00
T44				60.53	16.46	23.01	Yes	100.00
T45				33.11	50.34	16.52	Yes	100.00
T46				16.54	50.36	33.07	Yes	100.00
T47				42.16	47.22	10.61	Yes	100.00
T48				61.39	8.77	29.84	Yes	100.00
T49				32.98	53.40	13.61	Yes	100.00
T50				13.71	53.42	32.87	Yes	100.00
T51				11.74	35.99	52.27	Yes	100.00
T52				51.55	9.35	59.10	Yes	99.94
T53				18.15	49.72	32.12	Yes	100.00
T54				17.21	30.54	52.85	Yes	99.97
T55				27.10	35.67	37.19	Yes	99.99
T56				53.06	8.66	38.28	Yes	99.85
T57				15.81	33.62	50.57	Yes	100.00
T58				51.96	30.57	17.46	Yes	100.00
T59				18.98	38.86	42.15	Yes	100.00
T60				48.77	26.52	24.70	Yes	96.42

The accuracy of the plant disease detection system was calculated for different categories, demonstrating its effectiveness in identifying healthy and diseased plants. For the healthy plant category, the system correctly identified 19 out of 20 images, resulting in an accuracy of $\left(\frac{20-1}{20}\right) x100\% = 95\%$. For the grape black rot and blight leaf categories, the system achieved perfect accuracy, correctly identifying all 20 images in each category, yielding 100\% accuracy for both. When considering the overall performance across all categories, the system correctly classified 59 out of 60 images, leading to an overall accuracy of $(\frac{60-1}{60})$ x 100%=98.33%. These results highlight the robustness and reliability of the detection model in accurately distinguishing between healthy plants and various diseases.

Table 4.4 compares the feature extraction results for “Healthy”, “Grape Black Rot”, and “Blight Leaf” samples, highlighting differences in texture and statistical characteristics. Healthy leaves show low contrast, high homogeneity, and higher energy, indicating smooth and uniform textures. Grape Black Rot leaves have higher contrast, lower energy, and higher variance, reflecting more texture variation and less uniformity. Blight Leaf features lie in between, with moderate contrast and homogeneity. All leaves maintain high smoothness, but differences in entropy, kurtosis, and skewness reflect variations in intensity distribution across the leaf types. These features help distinguish between healthy and diseased leaves.

Table 4.4 Feature Extraction

Type of leaf	Feature Extraction
Healthy	Number of Features Extracted: 13 Feature Names and Values: Contrast: 0.2331 Correlation: 0.9119 Energy: 0.2744 Homogeneity: 0.9073 Mean: 152.7260 Standard Deviation: 43.6710 Entropy: 7.1947 RMS: 156.8352 Variance: 1160.2014 Smoothness: 1.0000 Kurtosis: 2.1742 Skewness: -0.2558 IDM: 255.0000
Grape Black Rot	Number of Features Extracted: 13 Feature Names and Values: Contrast: 0.5153 Correlation: 0.9210 Energy: 0.0825 Homogeneity: 0.8124 Mean: 115.1277 Standard Deviation: 63.6919 Entropy: 7.6718 RMS: 129.2402 Variance: 3074.8836 Smoothness: 1.0000 Kurtosis: 1.6134 Skewness: -0.1310 IDM: 255.0000

Blight Leaf	Number of Features Extracted: 13 Feature Names and Values: Contrast: 0.2543 Correlation: 0.8402 Energy: 0.2206 Homogeneity: 0.8818 Mean: 121.5504 Standard Deviation: 37.7680 Entropy: 7.0464 RMS: 125.0610 Variance: 734.0198 Smoothness: 1.0000 Kurtosis: 2.8288 Skewness: -0.8282 IDM: 255.0000

Table 4.5 summarizes the dataset composition and the predicted confidence levels for a plant disease detection model. The training dataset consists of 200 healthy samples, 150 samples with grape black rot, and 150 samples with leaf blight. The validation/test dataset includes 20 samples for each category. The model demonstrates high predicted confidence in its classification performance, achieving 100% confidence for healthy samples and 99.8% confidence for both grape black rot and leaf blight. This indicates the model is highly reliable in detecting and classifying these conditions.

Table 4.5 Comparison Table

	Healthy	Grape Black Rot	Leaf Blight
Train dataset	200	150	150
Valid/test dataset	20	20	20
Predicted confidence (%)	100.00	99.8	99.8

Figure 4.7 shows the relationship between the frequency of the training dataset (x-axis) and the predicted confidence percentage (y-axis). It highlights that as the frequency of the training dataset increases, the model's confidence in its predictions also increases. Specifically, at a frequency of approximately 150, the confidence is around 99.8%, and as the frequency rises to 200, the predicted confidence reaches 100%. The trend indicates a positive correlation where a larger training dataset improves prediction confidence, demonstrating that the model benefits from more data to improve its reliability.

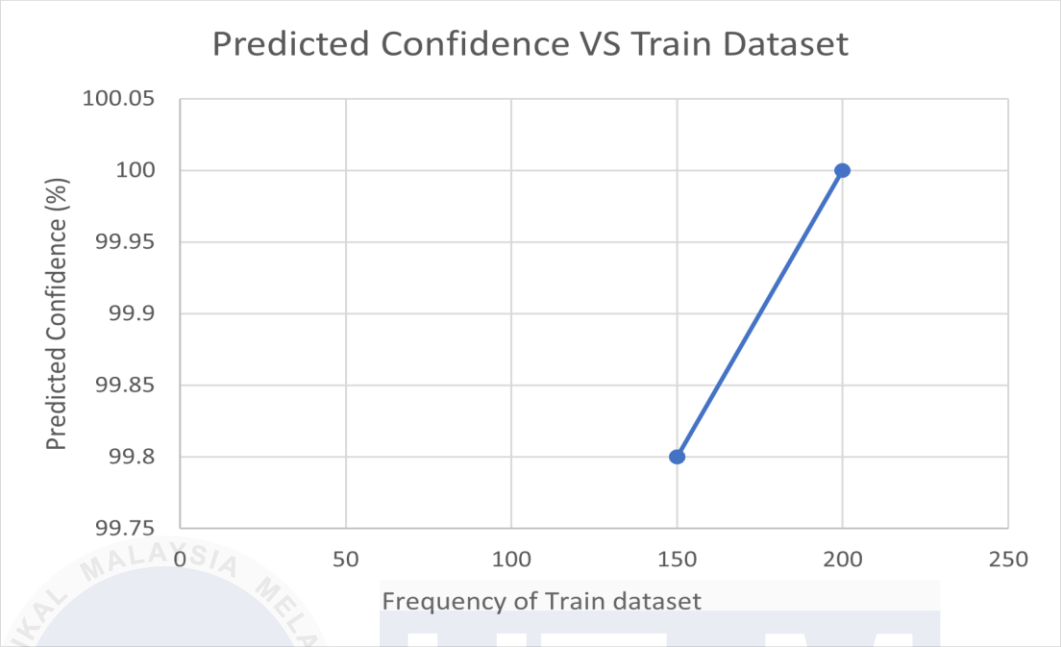


Figure 4.7 Graph Predicted Confidence vs frequency of train dataset

Figure 4.8 shows grape leaves classified into three categories: healthy, affected by Grape Black Rot, or suffering from blight, with each image labeled alongside a confidence percentage. The classifications likely come from a machine learning model trained to identify these conditions based on visual features. Healthy leaves show no visible damage, while diseased leaves exhibit distinct symptoms of their respective conditions. The purpose is to demonstrate the model's ability to accurately differentiate between healthy and diseased grape leaves.

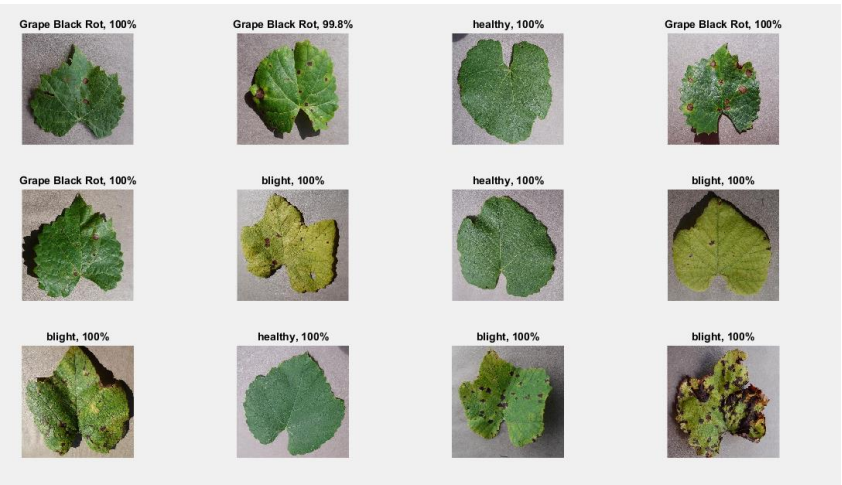


Figure 4.8 Predicted confidence results

4.3 Summary

This study investigates the results and discussions of the research on plant disease detection using CNN. It highlights the model's high accuracy in identifying and classifying various plant diseases, demonstrating its effectiveness compared to traditional methods. The chapter discusses the positive impact of preprocessing techniques on model performance and emphasizes the advantages of using deep learning for real-time disease detection in agriculture. Overall, the findings suggest that integrating CNN technology can significantly aid farmers in early disease identification and management, enhancing agricultural practices.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

In conclusion, the project on plant disease detection using CNNs within the MATLAB framework has made significant progress towards achieving its objectives. The project has successfully implemented advanced deep learning techniques and robust image processing capabilities to develop a highly effective solution for identifying and classifying plant diseases with remarkable accuracy.

The structured approach to project planning, as outlined in the methodology section, has ensured thoroughness and efficiency in the project development process. Starting from project initialization, defining scope and objectives, requirement analysis, design and planning, implementation, testing and validation, deployment, and maintenance and evaluation, each phase has been carefully executed to move the project forward systematically.

By leveraging the power of CNNs alongside MATLAB's versatility in algorithm development and data processing, the project has created a sophisticated tool that addresses critical challenges in agriculture. The future works mentioned in the conclusion section indicate a commitment to further enhancing the software functionality, demonstrating a forward-looking approach to continuous improvement and innovation.

Overall, the project has made significant strides in utilizing cutting-edge technology to revolutionize crop management practices, offering timely and precise disease detection that holds immense promise for enhancing agricultural productivity and sustainability. The adherence to a structured methodology and the successful integration of advanced techniques

highlights the project's dedication to achieving its objectives and delivering impactful results in the field of plant disease detection.

5.2 Potential for Commercialization

The project presents significant potential in addressing the pressing challenges of plant diseases that threaten agricultural productivity and food security. With the increasing incidence of such diseases, there is a growing demand for effective detection solutions among farmers, agricultural businesses, and researchers. By providing a reliable tool for early disease identification, this project not only aims to reduce crop losses but also to enhance overall agricultural yields, making it an asset in the agricultural sector.

Scalability is a key feature of the CNN-based plant disease detection system. Initially focused on grape leaves, the technology can be expanded to include a wider variety of crops, thereby broadening its applicability across different agricultural sectors. By enhancing the dataset and refining the model, the system can cater to diverse plant species, increasing its market appeal and potential for widespread adoption among farmers and agricultural professionals.

Commercial opportunities abound for this project, particularly through the development of a user-friendly mobile application. This app would enable farmers to upload images of their plants for instant disease diagnosis and treatment recommendations. Additionally, partnerships with agricultural suppliers could facilitate a marketplace within the app, providing users with direct access to necessary products such as pesticides and fungicides. Furthermore, offering expert consultation services through the app could create an additional revenue stream, ultimately promoting sustainable agricultural practices and contributing to global food security.

5.3 Future Works

To enhance the usability and accessibility of the CNN-based plant disease detection system, future work will focus on developing a marketplace platform tailored to farmers, gardeners, and agricultural stakeholders. The following features will be prioritized:

i) Consider effect of unbalance and harmonic to TL. Mobile Integration: Develop a user-friendly mobile application that integrates the CNN- based detection model, enabling users to upload leaf images and receive disease diagnosis and remedies instantly.

ii) Data-Driven Insights: Create a centralized dashboard where farmers can track disease outbreaks across regions, forecast risks, and access actionable insights for crop management.

iii) Instant Diagnosis: Enable users to upload images of their plants for immediate disease identification and treatment recommendations, complete with confidence scores.

REFERENCES

[1] R. Tripathi, “A Deep Learning Approach for Plant Material Disease Identification,” IOP Conf. Ser. Mater. Sci. Eng., vol. 1116, p. 12133, 2021, doi: 10.1088/1757-899X/1116/1/012133.

[2] M. Sethi, “Plant Disease Detection using Image Segmentation,” Int. J. Ayurveda Herb. Res., vol. 1, no. 1, pp. 15–18, 2023, doi: 10.54060/ijahr.v1i1.3.

[3] P. Srivastava, K. Mishra, V. Awasthi, V. Kumar Sahu, and P. Kumar Pal, “Plant Disease Detection Using Convolutional Neural Network,” Int. J. Adv. Res., vol. 9, no. 01, pp. 691–698, 2021, doi: 10.21474/ijar01/12346.

[4] P. Dhiman, A. Kaur, Y. Hamid, E. Alabdulkreem, H. Elmannai, and N. Ababneh, “Smart Disease Detection System for Citrus Fruits Using Deep Learning with Edge Computing,” Sustain., vol. 15, no. 5, 2023, doi: 10.3390/su15054576.

[5] M. Bouni, B. Hssina, K. Douzi, and S. Douzi, “Impact of Pretrained Deep Neural Networks for Tomato Leaf Disease Prediction,” J. Electr. Comput. Eng., vol. 2023, 2023, doi: 10.1155/2023/5051005.

[6] A. S. Zamani et al., “Performance of Machine Learning and Image Processing in Plant Leaf Disease Detection,” J. Food Qual., vol. 2022, 2022, doi: 10.1155/2022/1598796.

[7] M. V. Shewale and R. D. Daruwala, “High performance deep learning architecture for early detection and classification of plant leaf disease,” J. Agric. Food Res., vol. 14, no. June, p. 100675, 2023, doi: 10.1016/j.jafr.2023.100675.

[8] H. Amin, A. Darwish, A. E. Hassanien, and M. Soliman, “End-to-End Deep Learning Model for Corn Leaf Disease Classification,” IEEE Access, vol. 10, pp.31103–31115, 2022, doi: 10.1109/ACCESS.2022.3159678.

[9] E. Moupojou et al., “FieldPlant: A Dataset of Field Plant Images for Plant Disease Detection and Classification With Deep Learning,” IEEE Access, vol. 11, no. April,pp. 35398–35410, 2023, doi: 10.1109/ACCESS.2023.3263042.

[10] J. F. Restrepo-Arias, J. W. Branch-Bedoya, and G. Awad, “Plant Disease Detection Strategy Based on Image Texture and Bayesian Optimization with Small Neural Networks,” Agric., vol. 12, no. 11, pp. 1–18, 2022, doi: 10.3390/agriculture12111964.

[11] W. Gao, Z. Xiao, and T. Bao, “Detection and Identification of Potato-Typical Diseases Based on Multidimensional Fusion Atrous-CNN and Hyperspectral Data,” Appl. Sci., vol. 13, no. 8, 2023, doi: 10.3390/app13085023.

[12] K. Neupane and F. Baysal-Gurel, “Automatic identification and monitoring of plant diseases using unmanned aerial vehicles: A review,” Remote Sens., vol. 13, no. 19, 2021, doi: 10.3390/rs13193841.

[13] X. Chen, “The Study for Convolutional Neural Network and Corresponding Applications,” Theor. Nat. Sci., vol. 5, no. 1, 2023, doi: 10.54254/2753-8818/5/20230387.

[14] D. H. Hubel and T. N. Wiesel, “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex,” J. Physiol., vol. 160, no. 1, 1962, doi: 10.1113/jphysiol.1962.sp006837.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” Commun. ACM, vol. 60, no. 6, 2017, doi: 10.1145/3065386.

[16] K. Fukushima, “Artificial Vision by Deep CNN Neocognitron,” IEEE Trans. Syst. Man, Cybern. Syst., vol. 51, no. 1, 2021, doi: 10.1109/TSMC.2020.3042785.

[17] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, 2017, doi: 10.1109/TPAMI.2016.2577031.

[18] C. Y. Cao, J. C. Zheng, Y. Q. Huang, J. Liu, and C. F. Yang, “Investigation of a promoted you only look once algorithm and its application in traffic flow monitoring,” Appl. Sci., vol. 9, no. 17, 2019, doi: 10.3390/app9173619.

[19] W. Liu et al., “SSD: Single shot multibox detector,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2016. doi: 10.1007/978-3-319-46448-0_2.

APPENDICES

Appendix A Training Process Code

```
%% Training Process Based on Deep Learning Model Using AlexNet

clear; close all; clc;

% Step 1: Load and Preprocess the Data
% Replace 'dataFolder' with the path to your dataset folder
dataFolder = 'C:\Users\USER\Downloads\psm';
imds = imageDatastore(dataFolder, ...
    'IncludeSubfolders', true, ...
    'LabelSource', 'foldernames');

% Split the data into training and validation sets
[imdsTrain, imdsValidation] = splitEachLabel(imds, 0.7, 'randomized');
numTrainImages = numel(imdsTrain.Labels);
inputSize = [227 227]; % AlexNet default input size

% Resize images to match AlexNet input size and apply data augmentation
imageAugmenter = imageDataAugmenter( ...
    'RandXReflection', true, ...
    'RandXScale', [0.9 1.1], ...
    'RandYScale', [0.9 1.1]);
augmentedTrain = augmentedImageDatastore(inputSize, imdsTrain,
'DataAugmentation', imageAugmenter);
augmentedValidation = augmentedImageDatastore(inputSize, imdsValidation);

% Display sample training images
figure('Name', 'Sample Training Images');
for i = 1:9
    subplot(3, 3, i);
    I = readimage(imdsTrain, randperm(numTrainImages, 1));
    imshow(I);
end

% Step 2: Load Pre-trained AlexNet and Modify for New Classes
net = alexnet; % Load AlexNet
layersTransfer = net.Layers(1:end-3); % Keep all layers except the last 3

% Modify the last fully connected layer and classification layer
numClasses = numel(categories(imdsTrain.Labels));
layers = [
    layersTransfer
    fullyConnectedLayer(numClasses, 'WeightLearnRateFactor', 10,
'BiasLearnRateFactor', 10)
    softmaxLayer
    classificationLayer];

% Step 3: Set Training Options
options = trainingOptions('sgdm', ...
    'MiniBatchSize', 32, ...
    'MaxEpochs', 10, ...
    'InitialLearnRate', 1e-4, ...
```

```
    'ValidationData', augmentedValidation, ...
    'ValidationFrequency', 5, ...
    'Verbose', false, ...
    'Plots', 'training-progress');

% Step 4: Train the Network
netTransfer = trainNetwork(augmentedTrain, layers, options);

% Save the trained model
save('PDC_Train.mat', 'netTransfer');
disp('Network model was trained and saved as PDC_Train.mat');

% Step 5: Evaluate the Model
% Calculate the accuracy on the validation set
[YPred, scores] = classify(netTransfer, augmentedValidation);
YValidation = imdsValidation.Labels;
accuracy = mean(YPred == YValidation);
fprintf('Validation accuracy: %.2f%%\n', accuracy * 100);

% Display a few sample predictions
idx = randperm(numel(imdsValidation.Files), 4);
figure('Name', 'Sample Predictions');
for i = 1:4
    subplot(2, 2, i);
    I = readimage(imdsValidation, idx(i));
    imshow(I);
    label = YPred(idx(i));
    title(sprintf("Predicted: %s", string(label)));
end
```

Appendix B Function Code

```
function FINAL
    % Main GUI Figure
    fig = figure('Name', 'Plant Disease Detection', 'NumberTitle', 'off', ...
                 'Position', [100, 100, 1200, 700], 'MenuBar', 'none', ...
                 'Resize', 'off', 'Color', [0.9, 0.95, 1]);

    % Variables for storing data
    global I netTransfer inputSize axImage axAffection axPrediction axDisease
axSegment popupCluster segmented_images totalPixels YPred;
    inputSize = [227, 227, 3]; % AlexNet input size

    % Load Trained Model
    data = load('PDC_Train.mat'); % Load into a structure
    if isfield(data, 'netTransfer')
        netTransfer = data.netTransfer; % Extract the trained network
        msgbox('Trained Network Model Was Loaded', 'Model Loaded');
    else
        errordlg('The required variable "netTransfer" was not found in
PDC_Train.mat.', 'Error');
        return;
    end

    % Title Label
    uicontrol('Style', 'text', 'String', 'Plant Disease Detection System',
...
              'FontSize', 20, 'FontWeight', 'bold', 'BackgroundColor', [0.7,
0.85, 1], ...
              'ForegroundColor', 'k', 'Position', [400, 650, 400, 40], ...
              'HorizontalAlignment', 'center');

    % Axes for displaying images and results
    axImage = axes('Parent', fig, 'Units', 'pixels', ...
                   'Position', [50, 400, 300, 200], 'Box', 'on');
    title(axImage, 'Leaf Image', 'FontWeight', 'bold', 'FontSize', 12);

    axAffection = axes('Parent', fig, 'Units', 'pixels', ...
                       'Position', [400, 400, 300, 200], 'Box', 'on');
    title(axAffection, 'Affected Area (%)', 'FontWeight', 'bold', 'FontSize',
12);
    axis(axAffection, 'off');

    axPrediction = axes('Parent', fig, 'Units', 'pixels', ...
                        'Position', [50, 150, 300, 200], 'Box', 'on');
    title(axPrediction, 'Prediction Confidence (%)', 'FontWeight', 'bold',
'FontSize', 12);
    axis(axPrediction, 'off');

    axDisease = axes('Parent', fig, 'Units', 'pixels', ...
                     'Position', [400, 150, 300, 200], 'Box', 'on');
    title(axDisease, 'Disease Type', 'FontWeight', 'bold', 'FontSize', 12);
    axis(axDisease, 'off');

    % Axes for displaying segmented ROI
    axSegment = axes('Parent', fig, 'Units', 'pixels', ...
```

```
                    'Position', [750, 330, 300, 200], 'Box', 'on');
    title(axSegment, 'Selected Cluster (ROI)', 'FontWeight', 'bold',
'FontSize', 12);

    % Popup menu for cluster selection
    popupCluster = uicontrol('Style', 'popupmenu', ...
                             'String', {'Select Cluster', 'Cluster 1',
'Cluster 2', 'Cluster 3'}, ...
                             'Position', [750, 550, 300, 30], ...
                             'FontSize', 12, 'Callback', @displayCluster);

    % Arrange buttons at the bottom
    buttonWidth = 150;
    buttonHeight = 40;
    buttonSpacing = 30;
    totalWidth = (4 * buttonWidth) + (3 * buttonSpacing);
    startX = (1200 - totalWidth) / 2; % Center buttons horizontally
    startY = 20; % Fixed vertical position for buttons

    uicontrol('Style', 'pushbutton', 'String', 'Browse', ...
              'FontSize', 12, 'BackgroundColor', [0.7, 0.85, 1], ...
              'Position', [startX, startY, buttonWidth, buttonHeight], ...
              'Callback', @browseCallback);

    uicontrol('Style', 'pushbutton', 'String', 'Detect Disease', ...
              'FontSize', 12, 'BackgroundColor', [0.6, 0.8, 0.9], ...
              'Position', [startX + buttonWidth + buttonSpacing, startY,
buttonWidth, buttonHeight], ...
              'Callback', @detectionCallback);

    uicontrol('Style', 'pushbutton', 'String', 'Analyze Features', ...
              'FontSize', 12, 'BackgroundColor', [0.5, 0.75, 0.85], ...
              'Position', [startX + 2 * (buttonWidth + buttonSpacing),
startY, buttonWidth, buttonHeight], ...
              'Callback', @analysisCallback);

    uicontrol('Style', 'pushbutton', 'String', 'Remedy', ...
              'FontSize', 12, 'BackgroundColor', [0.6, 0.8, 0.7], ...
              'Position', [startX + 3 * (buttonWidth + buttonSpacing),
startY, buttonWidth, buttonHeight], ...
              'Callback', @remedyCallback);

    %% Callback Functions
    % Browse Image
    function browseCallback(~, ~)
        [filename, pathname] = uigetfile({'*.*'; '*.bmp'; '*.jpg'; '*.gif'},
'Pick a Leaf Image File');
        if filename == 0
            return;
        end
        I = imread([pathname, filename]);
        I = imresize(I, [256, 256]);
        totalPixels = numel(I(:, :, 1)); % Total number of pixels in the
image
        imshow(I, 'Parent', axImage);
        title(axImage, 'Loaded Image', 'FontWeight', 'bold');
    end

    % Detect Disease
```

```
    function detectionCallback(~, ~)
        if isempty(I)
            errordlg('Please load an image first.', 'Error');
            return;
        end

        % Resize for AlexNet
        Img1 = imresize(I, inputSize(1:2));

        % Classification
        [YPred, scores] = classify(netTransfer, Img1);
        predicted_score = max(scores) * 100;

        % Update Axes
        imshow(I, 'Parent', axImage);
        title(axImage, 'Detected Image', 'FontWeight', 'bold');

        axes(axPrediction);
        cla;
        text(0.5, 0.5, sprintf('%.2f%%', predicted_score), ...
             'Color', 'b', 'FontWeight', 'bold', 'FontSize', 16, ...
             'HorizontalAlignment', 'center');
        title(axPrediction, 'Prediction Confidence (%)', 'FontWeight',
'bold');

        axes(axDisease);
        cla;
        text(0.5, 0.5, char(YPred), ...
             'Color', 'r', 'FontWeight', 'bold', 'FontSize', 16, ...
             'HorizontalAlignment', 'center');
        title(axDisease, 'Disease Type', 'FontWeight', 'bold');
    end

    % Analyze Features
    function analysisCallback(~, ~)
        if isempty(I)
            errordlg('Please load an image first.', 'Error');
            return;
        end

        % K-means segmentation
        lab_he = rgb2lab(I);
        ab = double(lab_he(:, :, 2:3));
        nrows = size(ab, 1);
        ncols = size(ab, 2);
        ab = reshape(ab, nrows * ncols, 2);
        nColors = 3;
        [cluster_idx, ~] = kmeans(ab, nColors, 'distance', 'sqEuclidean',
'Replicates', 3);
        pixel_labels = reshape(cluster_idx, nrows, ncols);
        segmented_images = cell(1, nColors);
        rgb_label = repmat(pixel_labels, [1, 1, 3]);
        for k = 1:nColors
            colors = I;
            colors(rgb_label ~= k) = 0;
            segmented_images{k} = colors;
        end
    end
```

```
    % Display Cluster and Calculate Affected Area
    function displayCluster(~, ~)
        selectedCluster = popupCluster.Value - 1; % Get selected cluster
index
        if selectedCluster >= 1 && selectedCluster <= 3
            clusterMask = segmented_images{selectedCluster}(:, :, 1) > 0;
            if strcmpi(char(YPred), 'Healthy')
                affectedArea = 0.00; % Healthy case, affected area is always
0
            else
                affectedArea = sum(clusterMask(:)) / totalPixels * 100;
            end

            imshow(segmented_images{selectedCluster}, 'Parent', axSegment);
            title(axSegment, sprintf('Selected Cluster %d (ROI)',
selectedCluster), 'FontWeight', 'bold');

            axes(axAffection);
            cla;
            text(0.5, 0.5, sprintf('%.2f%%', affectedArea), ...
                 'Color', 'g', 'FontWeight', 'bold', 'FontSize', 16, ...
                 'HorizontalAlignment', 'center');
            title(axAffection, 'Affected Area (%)', 'FontWeight', 'bold');
        else
            cla(axSegment);
            cla(axAffection);
            title(axSegment, 'Selected Cluster (ROI)', 'FontWeight', 'bold');
        end
    end

    % Remedy Callback Function
    function remedyCallback(~, ~)
        if isempty(I)
            errordlg('Please load an image and detect disease first.',
'Error');
            return;
        end

        % Get the detected disease type
        diseaseType = char(YPred); % Ensure YPred is converted to a string

        % File paths for remedies
        remedyFolder = 'C:\Users\USER\Downloads\REMEDY'; % Folder where
remedy files are located
        switch lower(diseaseType) % Match disease type
            case 'blight'
                remedyFile = fullfile(remedyFolder, 'BLIGHT REMEDY.txt');
            case 'grape black rot'
                remedyFile = fullfile(remedyFolder, 'BLACK ROT REMEDY.txt');
            otherwise
                errordlg('No remedy available for the detected disease.',
'Error');
                return;
        end

        % Check if the file exists and display it
        if isfile(remedyFile)
            % Open the remedy file in the default text editor
            winopen(remedyFile);
```

```
        else
            errordlg(['The remedy file "' remedyFile '" does not exist.'],
'Error');
        end
    end
end
```