OPTIMISING POST-HARVEST FARM YIELD THROUGH UTILIZATION OF A MACHINE LEARNING BASED-FRUIT DISEASE CLASSIFICATION MODEL

By

Mbulelo Siyabonga Perfect Ngongoma

Student No: 22290726

A thesis submitted in fulfillment of the requirements for the Doctor of Engineering Degree in the Department of Electrical Power Engineering, Faculty of Engineering and the Built Environment

Durban University of Technology

March 2023

Supervisor: Dr. Musasa Kabeya

Co-Supervisor: Prof. Katleho Moloi

DECLARATION

I hereby declare that this dissertation is my work, and each text has been correctly referenced or cited. Moreover, this work has not been previously published in portion or whole for another degree at any other University.

This research was duly supervised by Dr. Musasa Kabeya and Prof. Katleho Moloi at the Durban University of Technology.

Submitted by: 01-Oct-2023

Student: Mr Mbulelo S.P. Ngongoma

Approved for Final Submission by:

Supervisor: Dr. Musasa Kabeya _

Co-Supervisor: Prof. Katleho Moloi

DEDICATION

This research is dedicated to the memory of my late mother, **Ms. Bonakele Monica Ngongoma**. May her soul rest in peace. I would also like to dedicate this work to my fiancée, my kids and my grandfather, Themba Alois Ngongoma, they have been my source of strength.

ACKNOWLEDGMENT

I would like to express my gratitude to my supervisors, Dr. Musasa Kabeya and Prof. Katleho Moloi, for offering excellent support during this research study. I also want to express my gratitude to my family for their continuous support and encouragement throughout this research study.

ABSTRACT

This research study proposed 4 key improvements to the classical fruit disease detection models which have been proven to increase their classification and accuracy levels. The globe and, more particularly, the economically developed regions of the world are currently in the era of the Fourth Industrial Revolution (4IR). Conversely, the economically developing regions in the world (and more particularly, the African continent) have not yet even fully passed through the Third Industrial Revolution (3IR) wave. Moreover, Africa's economy is still heavily dependent on the agricultural field. On the other hand, the state of global food insecurity is worsening on an annual basis due to exponential growth in the global human population, which continuously heightens food demand in both quantity and quality. This justifies the significance of the focus on digitizing agricultural practices to improve farm yield to meet the global steep food demand and stabilize the economies of the African continent and countries such as India that are largely dependent on the agricultural sector for revenues. Technological advances in precision agriculture are already improving farm yields, especially in the more economically developed regions of the globe, although several opportunities for further improvement still exist. Hence, this study evaluated a particular area of precision agriculture, the plant disease detection models which fall under decision support systems. The aim was to gauge the status of the research in this field, identify opportunities for further research, and propose technical amendments to the traditional plant disease detection models to improve their functional efficiency and accuracy.

Hence, through reviewing the available literature, this study has realized the dearth of literature focused on the real-time monitoring of the onset signs of diseases before they spread throughout the whole plant. There is also substantially less focus on real-time mitigation measures such as actuation operations, spraying pesticides, spraying fertilizers, etc., once a disease is identified. Very little research has focused on the combination of monitoring and phenotyping functions into one model capable of multiple tasks. Most of the proposed plant

disease classification models are based on a 2-Dimensional ‘view’ of the sample, which might pose challenges in the case of spherical or cylindrical plant samples such as fruits.

Therefore, four key proposals were made in this research study. Proposal 1 was an improved image pre-processing technique for Machine Learning-based plant disease classification models. This technique dissolves a Red-Green-Blue (RGB) image into individual red, green and blue planes and performs the thresholding process on one or more planes and superimposes the resulting binary images. This has proven to yield better feature segmentation depending on the application. Proposals 2 and 3 aimed to grant a classification model a ‘3-Dimensional view’ of the sample to eliminate any ‘blind spot’ which might be hiding important features that would directly impact the classification decision had they not been hidden. Proposal 2 achieved this by using multiple input image cameras, while Proposal 3 employed a revolving sample stand that allows a single input camera to take multiple input images (at different angles) from the sample. Proposals 2 and 3 were tested in classifying healthy from black rot-affected oranges, and they outperformed the traditional plant disease detection model by classifying correctly even the oranges with small and uneven distribution of black rot.

Proposal 4 combined crucial processes that are traditionally stand-alone in farming operation into a single hybrid model, hence the Hybrid Fruit Disease-Quality Monitoring and Sorting Model. This model offers post-harvest benefits and has also been conceptualized based on fruit plant samples. This model could detect diseases; perform quality checks (punchers, skin pills, etc.); perform grading based on these quality checks; and sort the fruits into designated bins according to their assigned class. It aims to lower the price of digital farming technology and avail it to low-budget farms. This model was tested on oranges (healthy, black rot-affected, and generally damaged) and apples (healthy, botch-affected, and generally damaged). The model managed to classify each of these diseases; perform the quality check based on the general fruit damages; and sort/grade (conceptually) these fruits according to these different classes. Its classification accuracy was 100% since all the test samples were classified correctly.

Although proposals 3 and 4 of this study were electromechanical systems, their modelling and testing have been limited to the electrical aspect. A full electromechanical design still needs to be implemented for a full capability study to be done in practical settings. Another limitation of this study was collecting the fruit samples with the desired disease symptoms distribution. The test samples were collected from the local vendors and the variety was greatly limited.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

Acronym	Description
AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CMOS	Complementary Metal–Oxide–Semiconductor
CCD	Charge-Coupled Device
DL	Deep Learning
GHI	Global Harvest Initiative
GNSS	Global Navigation Satellite System
H-DQMS	Hybrid Fruit Disease-Quality Monitoring and Sorting
ICT	Information And Communication Technology
k-NN	K-Nearest Neighbor
ML	Machine Learning
SVM	Space Vector Machine
TDI	Time Delay Integration
UVC	USB Video Class
WHO	World Health Organization
2-D	2-Dimensional

3IR	3rd Industrial Revolution
3-D	3-Dimensional
4IR	4th Industrial Revolution

CHAPTER 1: INTRODUCTION

Chapter 1 discusses the introductory aspects of this research study. The context, background and motivation of the study, problem statement, aims and objectives, research questions, study limitations and the list of publications are amongst the key aspects presented in this chapter.

1.2. Context

This thesis presents a research investigation into Machine Learning (ML)-based precision agricultural plant disease detection tools utilized in elite farms to inform early plant disease countermeasures. The study hence proposes several improved models that present improved functional reliability, accuracy, and efficiency. Crop diseases are a major root cause of losses in farm yields. Therefore, monitoring and the early identification of crop stress or more particular crop diseases, pests, nutrient deficiencies, and weeds are imperative in an effective farming operation [1-3]. In conventional agricultural practices, farmers rely upon visual observations of specimens to identify diseased leaves, fruits, roots, etc. [1-3]. However, this method faces several challenges ranging from the need for continuous observation of specimens which is tedious, to expensive costs for large farms, and a high level of inaccuracy [3]. Shruthi [3] asserts that agriculturalists often consult experts for the identification of infections on their crops, which incurs even more costs and have longer turnaround times. On the other hand, keeping track of different morphological characteristics of plants at different stages of their growth can help facilitate the early detection of a deviation from normal trends which may be an eminent disease at its conception [1, 3-5].

Therefore, all these limitations of classical farming methods, coupled with the tremendous pressure to keep up with an exponentially growing demand for food both in quantity and quality, have served as the push factors for researchers to devise new strategies and tools to digitize the agricultural field with the prime objective of increasing farm yields and produce [4, 6]. Hence, the similar mentioned premises are the prime motives that have inspired the undertaking of this research study.

1.3. Background and Motivation

Disease, pest or weed detection has been achieved by utilizing ML over the last two decades [6-12, 15, 17]. Padmavathi [16] defines ML as an intelligent technique where a machine is capacitated to recognize a pattern, recall historical information, and train itself without being commanded to do so. Both supervised and unsupervised training strategies can be utilized for machine training. Whilst there are distinct training and assessment datasets for supervised training, there is no such distinction for unsupervised training datasets. Kern [17] states that since ML is an evolving procedure, the machine's performance becomes better with time [14, 17]. As soon as the machine has finished learning or training, it may classify the data [18], make predictions [18, 19], and even generate fresh test data from which its re-trains itself and the process goes on and on [20]. Maddikunta [21] defines ML as a decision-making tool capable of visualizing the potentially complicated inter-relationships between important parameters and making educated predictions and/or decisions.

Maddikunta [21] further provides an illustration of ML application in precision agriculture as seen in Fig. 1.1. In the three-level precision agricultural layout shown below, the 1st level, which is the physical layer, represents all the field equipment such as sensors, trackers, actuators, probes, etc. [21, 22] that are in physical connection with the farm environment collecting data for further processing. In the second level, the edge layer is where the processing of the data collected in level 1 is taking place to convert the raw data into useful information which is used to inform decision-making [21, 22]. Decision-making takes place at this level through computational tools like computers, microcontrollers, microprocessors, etc. In the third level, the cloud layer, the storage of data for iterative training of the machine takes place [21, 22].

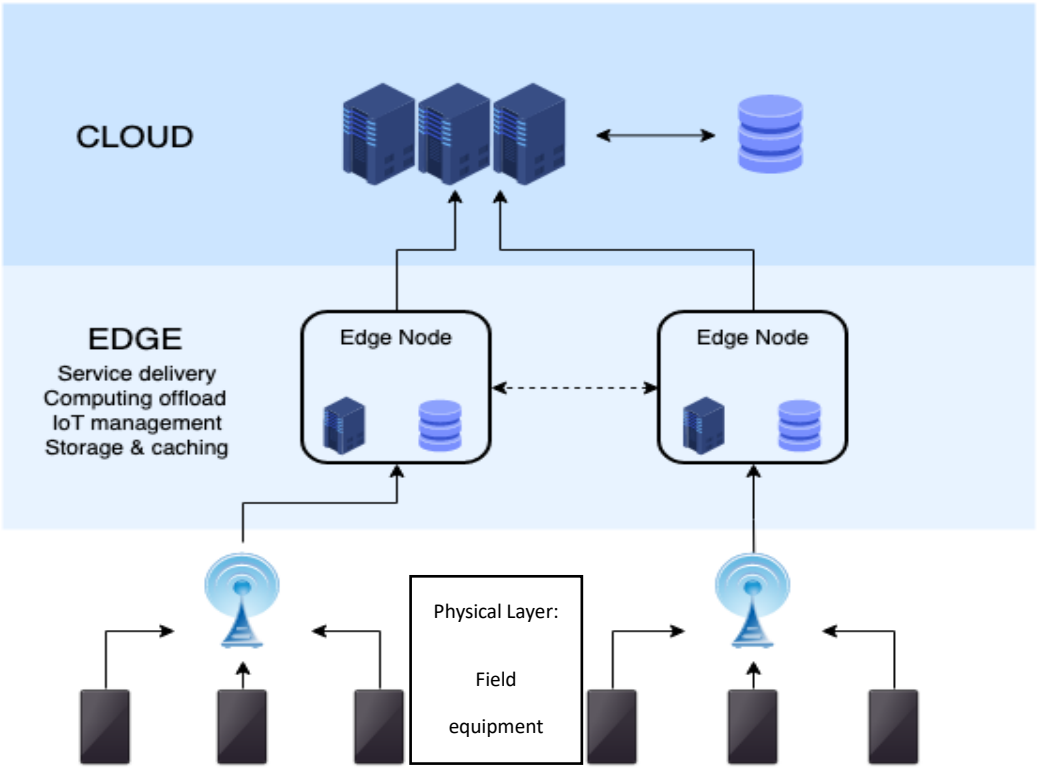


Figure 1-1: Example of ML in precision agriculture

Therefore, the ML-based plant disease detection models are generally made up of two main sub-systems, namely the image processing system and the classification system [23]. The image processing is further sub-divided into 5 steps, whilst there are also 5 most-cited classification protocols [21-23] summarized in Table 1.1.

Table 1-1: Summary of image processing steps and different classification techniques in plant disease detection

A typical general plant disease detection system	
Summary of image processing steps	*Different classification techniques*
• Image processing: ❖ Image acquisition ❖ Image pre-processing ❖ Image segmentation	• Support Vector Machine (SVM) • Artificial Neural Network (ANN) • K-Nearest Neighbour (KNN) • FUZZY Classifier

❖ Feature extraction	• Deep Learning

Recent studies on phenomics and high-throughput picture data gathering are available. However, most of the research on image interpretation and processing can be found in textbooks that delve into extensive detail about the methodologies [24, 25]. Fig. 1.2 illustrates the relationships between the image-processing steps shown in Table 1.1.

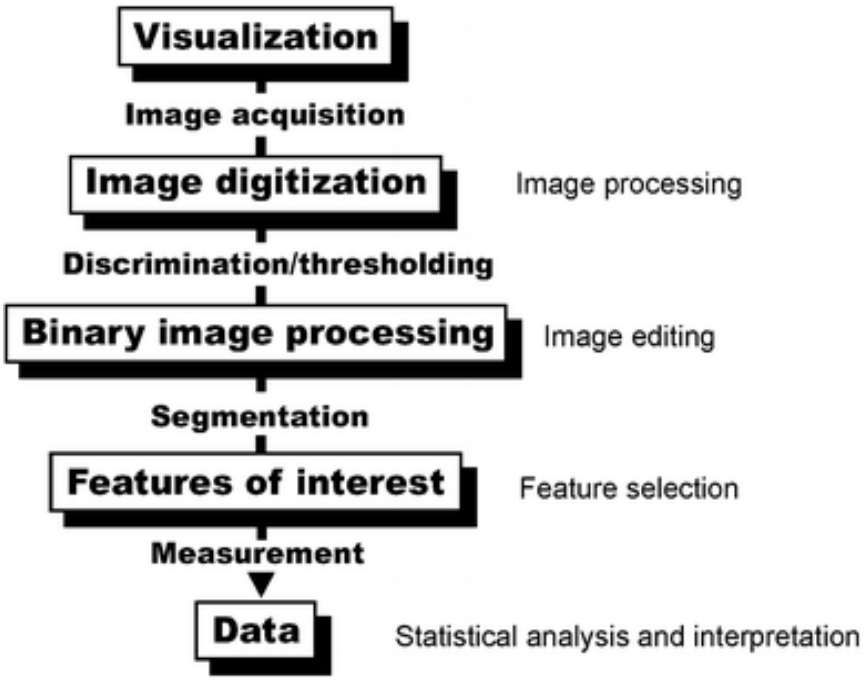


Figure 1-2: Steps of image processing

The rapid human growth over recent decades has resulted in an enlarged demand for agricultural foods, which in turn has led to a large expansion of farming. Pests and diseases are major obstacles to achieving this productivity outcome as crop output is continually threatened by diseases and insect pests. Therefore, it is very important to develop efficient methods for the automatic detection and classification of pests and diseases in crops. Several studies have been proposed on plant disease/pest/weed detection systems that employ the above-described general format. Noticeable in their trend is that they are growing in volume, complexity, and overall success in achieving their objectives. The following table summarizes the core studies that inspired this research:

Table 1-2: Summary of the literature review on ML-based plant disease/pest/weed detection

List of publications that inspired this research study				
Classification Method	*Plant/Crop*	*Reference*	*Number of diseases*	*Results*
ANN Classification	Potatoes	[14]	5	93.1% accuracy
	Pomegranate	[9]	4	90% accuracy
	Groundnut	[11]	4	94.71% accuracy
	Cucumber	[12]	2	81% accuracy
SVM Classification	Grapevine	[2]	2	88.89% accuracy
	Citrus fruits	[13]	2	95% accuracy
	Potato	[14]	2	90% accuracy
	Soybean	[8]	3	90% accuracy
	Oil palm	[17]	2	97% accuracy
	Tea	[16]	3	93% accuracy
CNN Classification	25 different crops	[2]	58	99.53% accuracy
	Peach, Cherry, Pear, Apple and Grapevine	[19]	13	96.3% accuracy
	14 different crops	[6]	26	99.35% accuracy
	Soybean	[4]	3	99.32% accuracy

Fuzzy Classification	Wheat	[5]	1	88% accuracy
KNN Classification	Sugarcane	[13]	1	95% accuracy
	Cotton	[20]	1	88% accuracy

1.4. Problem Statement

The ML-based decision support systems, more particularly the plant disease detection models have been a promising solution in battling with the losses incurred by plant diseases on farm yields [6-12], however there exists several opportunities to improve the efficiency and accuracy of this technology. They author has identified the following challenges faced with the current plant disease detection models:

- The plant disease detection models in the literature consulted by the author utilize single input image sensors or cameras. This implies that a classification model has a 'limited view' of a plant sample under evaluation especially in cases where the plant samples are spherical or cylindrical such as fruits, roots, flowers, etc. If a classification model can 'see' a limited surface area of a spherical or cylindrical sample where the disease symptoms are unevenly distributed (which is normally the case at the early stages of the infection [12]), then the classification model might/might not give an inaccurate classification depending on the distribution of the disease symptoms and orientation of the plant sample when the input image was captured. More details of this limitations will be discussed in Chapter 5.
- The image pre-processing techniques that have been proposed on the literature that the other have consulted are based either on the RGB or greyscale images. Individual Red, Green or Blue colour plane images seldom provides clear details on images and a precise distinction between the foreground and the background compared to both the RGB and greyscale images.
- The plant disease detection models in the literature consulted by the author are 'single purpose' machines. Though some have been designed to detect diseases on different plant species which others detect different diseases on a single plant species, the author did not

come across a hybrid model that has been designed to perform disease detection, grading and sorting operation that can be utilized for post-harvest benefits. The author believes that integrating such crucial operations into a single universal machine could help drive down the cost of this technology and make it accessible even to low budget farms in the African continent.

1.5. Aims and Objectives

This research study has the following aims and objectives:

- To develop deep learning-based plant disease detection model(s) that can 'see' the full 360° 3-dimensional view of a plant sample under evaluation such that the disease classification is made upon consideration of an image or images covering the full surface area of a sample as opposed to just the area exposed to the camera's light of sight.
- To develop and improved image pre-processing technique that is based on thresholding individual green, red and/or blue color planes as opposed to thresholding the raw RGB image.
- To develop a hybrid model that can detect plant diseases, preform harvest grading and sorting operations on a single station.

1.6. Research Question

This research study seeks to answer the following research questions:

- What topological amendments can be done on a classical plant disease detection model to enable it to 'see' or capture the full 360° surface of a plant sample under evaluation?
- How can enabling the plant disease detection to model a full 360° surface of a plant sample help improve its classification efficiency and accuracy?
- Can pre-processing the individual green, red or blue planes of a plant sample image results to a better subsequent feature segmentation compared to pre-processing the original RGB image?

- What measures can be done to lower the cost of automatic plant detection and similar so it can be accessible even to emerging farmers especially in the African continent?

1.7. List of Research Outputs

This section presents a list of research outputs that have emanated from this research study and their corresponding statuses.

Table 1-3: List of research outputs and corresponding statuses

List of research outputs from this research study		
Type of Research Output	*Topic*	*Status*
1. MDPI Applied Science Journal	A review of plant disease detection systems for farming applications	Published
2. AFRICON conference Article	An Improved Image Pre-processing Procedure for a Fruit Disease Detection Model	Accepted for publication
3. ELSEVIER Agricultural Systems Journal	Design And Modeling of A 3-Dimensional Image Input-Based Fruit Disease Detection Model Using Deep Learning Technology	Under Review
4. Springer Scientific Reports Journal	Design and Modeling of a Multi-camera-based Disease Detection Model	Under Review
5. POWER AFRICA conference article	Design and modeling of a Deep Learning Hybrid Fruit Disease Classification-Sorting Model	Under Review

6. MDPI Science Journal	Lighting Conditions Impact Evaluation on the Classification Accuracy and Efficiency of the Image-based Plant Disease Classification Model	Editing

1.8. Thesis Organization

This thesis employs the following structure:

- Chapter 1: Introduction to research work already done, motivation, research question(s), aims and objectives, thesis organization;
- Chapter 2: Literature review, detailed track record of the work done thus far on ML-based precision agriculture plant disease monitoring similar systems;
- Chapter 3: Methodology, selection of tools, methods, techniques, or algorithms used to achieve the objectives of the research;
- Chapter 4: Design and modelling of the traditional plant disease detection model (applied in fruit), analysis of results, and identifying as well as quantifying (where possible) the challenges;
- Chapter 5: Design and Modelling of the proposed solutions, evaluation of results, and comparison of the proposed to the traditional systems;
- Chapter 6: Overall conclusion on whether the study's objectives were met or not and why, and future recommendations.

1.9. Limitations of the Study

This study makes four proposals, two of which are electromechanical systems. The testing of all the proposals presented in this work will be limited to only the electrical aspects although all the mechanical aspects will be covered in the conceptualization phase. This is the case because the mechanical aspects are not directly part of the core contributions of this work.

Rather, they are part of the support systems around the main classification models. This study is also not intended to investigate the impact of fruit disease in different geographic locations but rather focuses on improving the methods utilized to detect fruit diseases to improve classification accuracy and efficiency. The study is also exempted from discussing the regression/prediction of disease occurrence probabilities, as its core objective is to develop methods to assist the traditional plat disease detection model to perform at an improved efficiency and accuracy. All the methods developed in this study have been tested on fruits (oranges and apples). However, they can be extended to other plant parts such as flowers, seeds, roots (e.g., potatoes), amongst others.

Several general assumptions have been made throughout this research and they have been made to clearly set the scope of this research study. These assumptions are stated below with their corresponding reasons:

- The first is the assumption that there exists a commercial farm that specializes in apples and oranges. This has been done to display the expansive range capability of a proposed classification model to cover the different stock-keeping units (SKU) of a given farm (different SKUs, in this case, mean different fruits).
- The second assumption is that these two species of fruits, the oranges and apples, suffer from two main diseases, black rot, and botch disease respectively. This condition has been set to display the capability of the proposed model to detect different fruit diseases. In practical conditions, numerous diseases and nutrient deficiencies can affect citrus fruits such as oranges, as well as apples, namely black-spot, cankers, sodium deficiency, potassium deficiency, etc., depending on the region and climate in which the farm exists [72]. However, it is beyond the scope and objectives of this research study to consider different diseases and nutrient deficiencies affecting each fruit. Hence, a single disease was assumed each for oranges and each for apples, and it should be noted that any diseases could have been chosen. The black rot and botch were selected based on their common nature in both these fruits and hence the abundance of image datasets covering them.
- The third assumption is that these fruits seldom sustain a range of physical damages during their life cycle. These include pest damage, fruit birds' pecking damage, weather-based damages such as acid rain, storm, or hail, as well as damages induced during the manual or

automated harvesting due tool manual tools utilized and/or automated machinery moving parts, respectively. This is a practical assumption, and a wide range of imagery sources exist that support the occurrences claimed by this assumption. This condition has been set to display the ability of a proposed classification model to perform the quality assurance checks and preliminary fruit grading by discriminating any fruits sustaining any physical damage as such damages serve as the onset point for bacterial infections.

- The fourth assumption is the possibility of different fruit SKUs mixing unintentionally. This study is aware of cases where fruit mixes occur deliberately and fruit products that feature fruit blends. However, it has also been clear that even with such products which intentionally require fruit blends, a specific ratio composition of raw fruit is required other than a random fruit mixture composition, which suggests that the original fruits ought to be separate at the beginning of the production process, and the mixing composition and operation must be predetermined and controlled respectively. Hence, fruit mixing prevention is imperative at the farm level. Fruit mixing is possible during manual and automated harvesting operations, as well as due to poor warehousing. This assumption serves to display the capability of the proposed model to perform the sorting operations at a faster rate and more reliably compared to manual sorting, which is widely used across developing and developed farms.

1.10. Summary

The introductory chapter outlined the background, context and motivation of the research study presented in this thesis. It further outlined the research questions as well as the research's aims and objectives. In Chapter 2, a detailed literature survey that was undertaken during this research study and the research opportunities identified are presented.

CHAPTER 2: LITERATURE REVIEW

2.1. Introduction

Chapter 2 discusses the tools required to implement a plant disease detection model and the recent research developments in this field. All the elements of a classical plant disease detection model, namely the image pre-processing techniques, image processing techniques and machine learning classification algorithms, are discussed. The literature review is also presented, whilst the research opportunities identified are discussed at the end of this chapter.

2.2. General Context

The recent research trends in precision agriculture, particularly in the disease/pest/weed detection area, are discussed for comprehending the Artificial Intelligence (AI) tools and scientific background required to implement these Machine Learning (ML)-based precision agriculture systems. The disease/pest/weed detection system is a focal point of this research study because diseases and pests cause the largest losses in farms [26].

This research study has realized that disease, weed, pest, nutrient deficiency, morphological feature and detection systems may all have similar working principles where a high-quality picture is acquired from a farm specimen, and an ML algorithm is then fed with such a picture, after processing and making a classification of different diseases. Therefore, these systems can have similar prototypic architectures and a farmer can have one universal robotic system that has a few change parts (such as cameras, sensors) and different software programs that are specific to different activities. This premise is one of the main ideas around which this study centered. Fig. 2.1 shows the overall structure of the rest of Chapter 2.

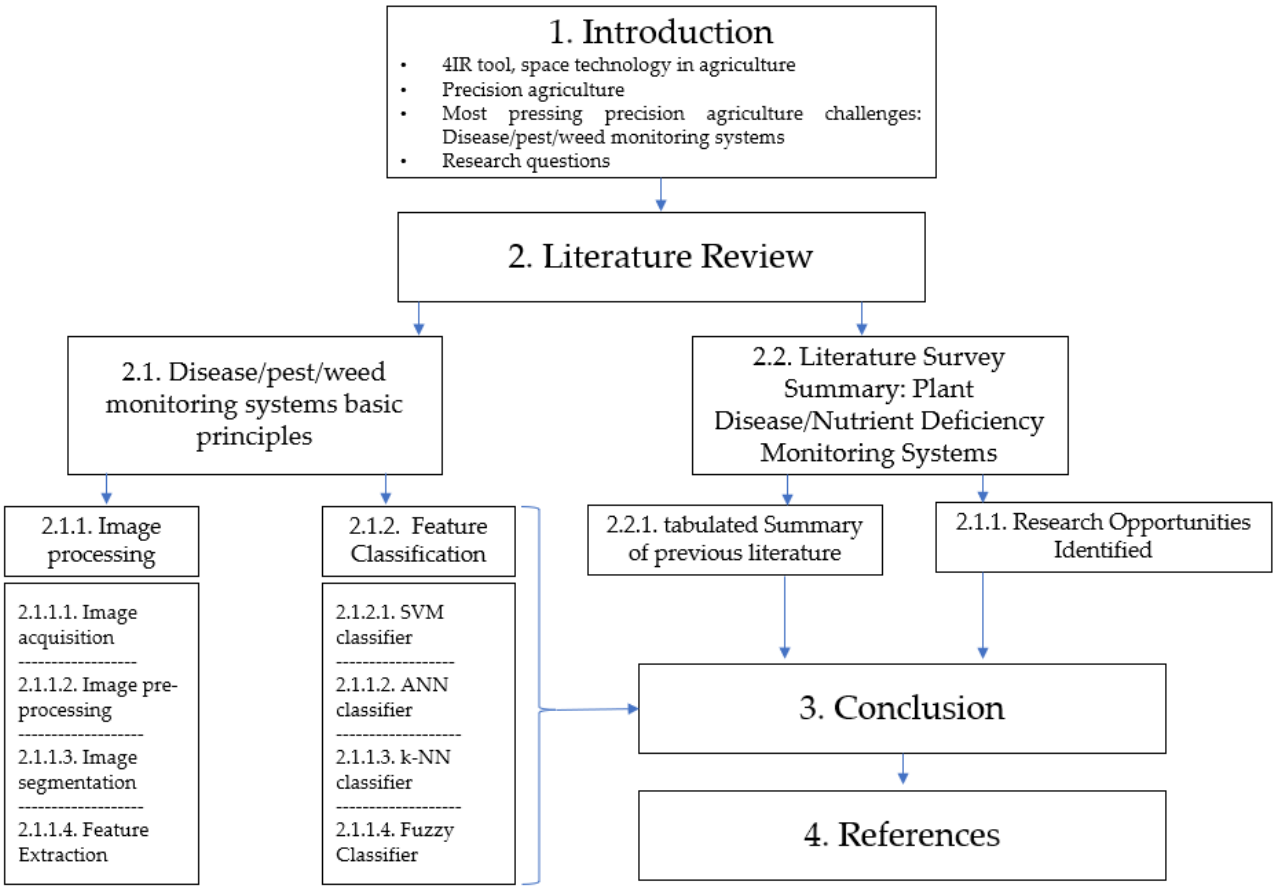


Figure 2-1: The general structure of the literature review

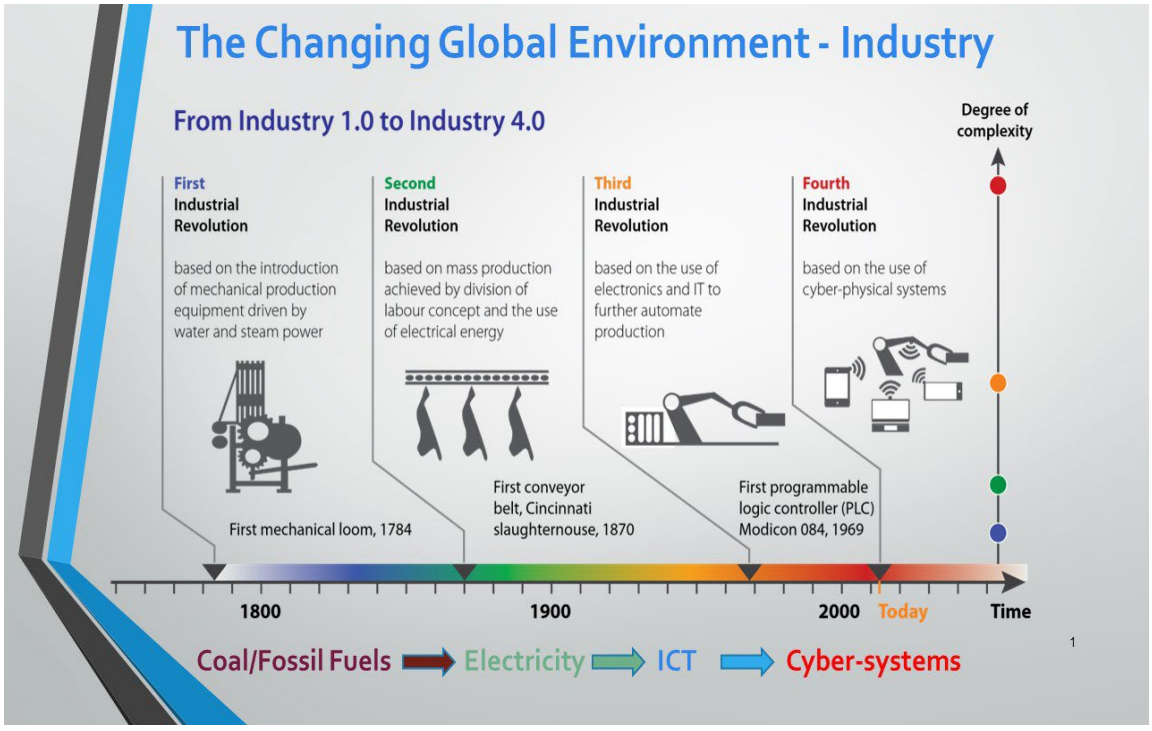


Figure 2-2: A historical perspective of the Changing Global Environment [DUT Inaugural Lecture – IED]

2.3. Literature Review: Precision Agriculture - Plant Disease Monitoring Systems Research Developments

As briefly stated in Chapter 1, the timely identification of plant diseases is imperative for a productive farm [23]. Manual farming operations such as vision-based plant disease monitoring systems are slow and inaccurate, which necessitate systems such as plant disease monitoring systems. In conventional agricultural practices, farmers rely upon visual observations of specimens to identify diseased leaves, fruits, roots and other parts of crops [23]. However, this method faces several challenges, such as the need for continuous checking and observation of specimens, which is tedious and expensive for large farms but most importantly very much less accurate [24-26]. Takada [27] asserts that agriculturalists often consult experts for the identification of infections on their crops, which incurs even more costs and has longer turnaround times. The earlier-stated limitations of classical farming methods coupled with the pressure to keep up with an exponentially growing demand for food both in quantity and quality have served as the push factors for researchers to devise new strategies and tools to digitize the agricultural field with the prime objective of increasing farm yields and produce [28]. The forthcoming sub-section discusses the general plant disease detection system, and one should note that the same general topology can be used to monitor pests, weeds, morphological features, and the like.

2.3.1. Plant Disease Detection System Basic Principles

Section 1.5 in Chapter 1 best outlines the basic principles of machine learning algorithms. With reference to Fig. 2.1, machine learning algorithms can be represented in 3 level architectures, the first, second and third levels being the physical, edge and cloud layers respectively. The physical layer comprises sensors and all the devices that collect the inputs for the algorithm [28]. The edge layer holds the actual machine learning model that performs the classification in the case of plant disease detection models [28, 117]. The cloud layer holds the training data and the historical performance information that is used to re-train the model and improve its classification efficiency [28, 117]. It is further stated in Section 1.5 that plant disease

detection systems generally have image processing and classification parts where the image processing can further be dissociated into image acquisition, image pre-processing, image segmentation and feature extraction, as in Table 1.1 and Fig. 2.3. The most cited machine learning classification algorithms in the application of disease detection systems are the SVMs, CNNs, k-NNs, Fuzzy Classifiers, and deep learning [17-19].

2.3.1.1. Image Processing Steps

Fig. 2.3 summarizes the steps for image processing that will subsequently be utilized by the ML-based classification algorithms for plant disease classification.

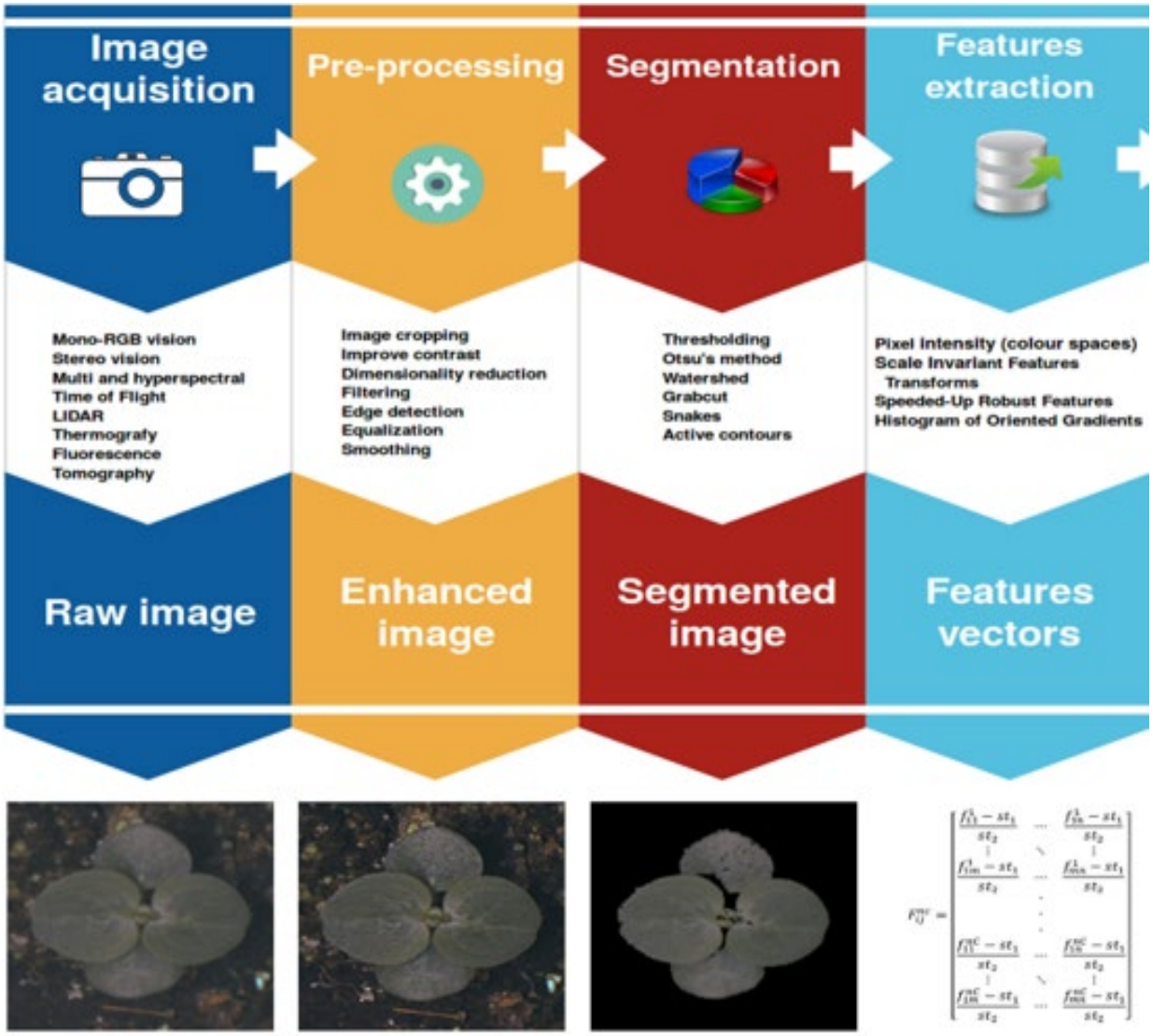


Figure 2-3: Steps of image processing

(i) Image Acquisition

Image collection is the first step of a system for detecting plant diseases [6, 8, 12]. Image sensors, scanners and unmanned aerial vehicles (UAVs) can all be used to capture photos of plants [23]. The commonly utilized image acquisition tools are a charge-coupled device (CCD) and complementary metal oxide semiconductor (CMOS) [23]. Both these camera technologies convert light signals and protons to digital data, which is then further transformed into a picture [27]. However, their methods of turning light signals into image data vary [16]. In a CCD camera, the light signals are transferred through a series of adjacent pixels before being amplified and converted into image data at the end of these pixel strings [17, 18]. This enables the CCD cameras to possess minimal degradation during the image acquisition process [19]. The CCD cameras generate sharp pictures with reduced distortion [18]. Contrarily, in CMOS cameras, the light signals are collected, amplified and converted at each pixel of the image sensor [15]. This enables the CMOS devices to generate images faster than CCD devices since each pixel can convert light signals into an image locally [17, 118]. CMOS devices are normally preferred in projects that are low budget since they are cheap compared to CCD devices, have lower power consumption, and can acquire high-quality images faster than their CCD counterparts [17-19]. Fig. 2.4 shows the serial versus localized pixel image conversion of CCD and CMOS image sensors respectively.

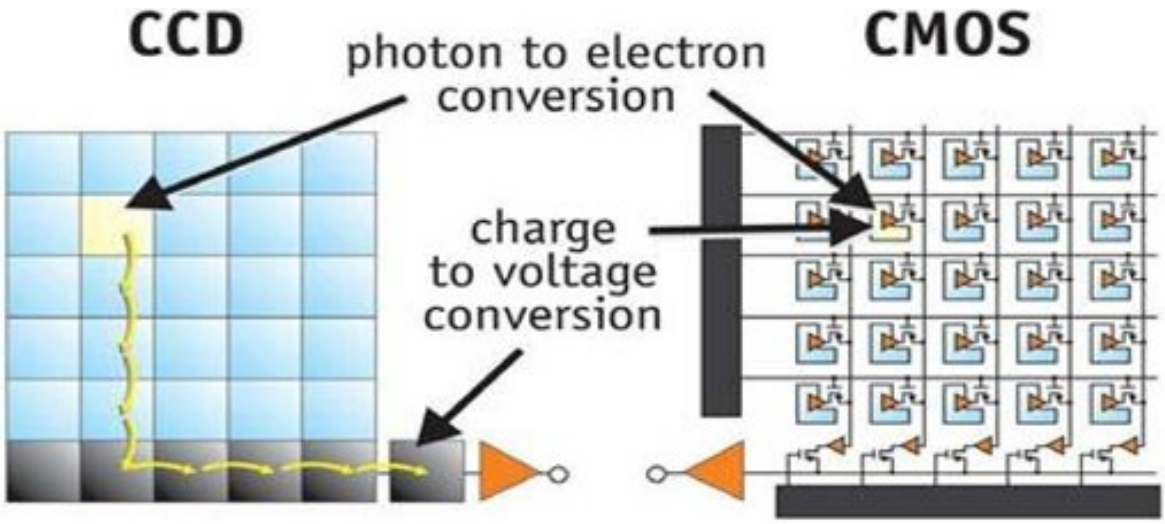


Figure 2-4: CCD vs. CMOS image conversion

An imaging acquisition tactic known as time delay and integration (TDI) can be combined with either CCD or CMOS technology to drastically improve their image acquisition capabilities [20]. Applications involving fast-moving objects, requiring high precision and the capacity to function in extremely dim lighting environments use TDI [20, 21]. Refer to Fig. 2.5 for an example of a high-speed application of TDI technology where a high-velocity train was captured with a normal and TDI-featured camera on the left and right pictures respectively. When the camera was operated in normal mode, the image of the train was a blur due to its high velocity and dim lighting conditions. However, the incorporation of a TDI mode countered these challenges and produced a clear detailed picture of a train.

Figure 2-5: Steps of image processing

After an image has been captured with a CCD or CMOS device, with or without TDI technology incorporated, a captured image should proceed to the following step of the image processing, which is normally image segmentation [3, 5, 11, 12, 16]. The segmentation of an image is a process whereby the features of interest are extracted from the rest of the image and irrelevant features are masked [10]. The features of interest are referred to as the foreground while the irrelevant ones are referred to as the background [16, 117]. The creation of the foreground versus background is dependent on picture properties like color, spectrum brightness, edge detection, and neighbor resemblance, to name a few [17, 118]. However, image

pre-processing may occasionally be necessary before an effective image segmentation can take place [3, 8, 11, 22].

(ii) Image Pre-processing

This is a crucial step in the ML-based disease detection system [14]. Pre-processing of an image deals with the correct setting of image contrast and filtration of interference signals resulting in noise and hence blurry images [18, 19]. This procedure can greatly enhance the precision of feature extraction and the correct disease detection in general [15, 119]. Pre-processing typically involves straightforward treatments like image cutting, clipping, cropping, filtering, trimming and de-blurring, amongst others [3]. Wang [23] explained a typical image pre-processing procedure that is generally employed in image-based detection systems, as shown in Fig. 2.6.

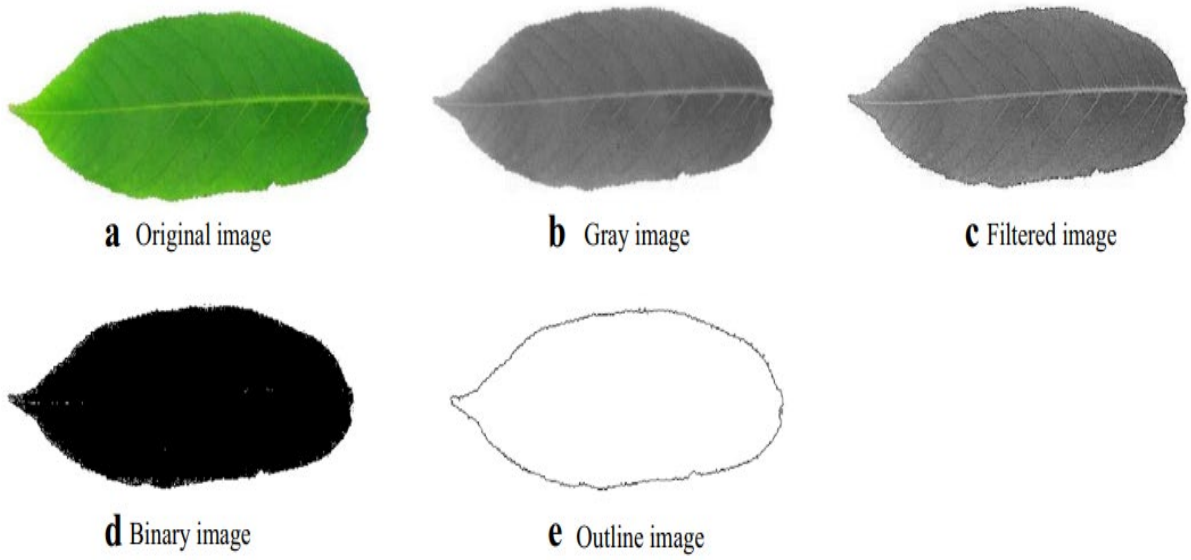


Figure 2-6: General pre-processing procedure for plant-based feature detection systems

The first step in the procedure illustrated in Fig. 2.6 involves the transformation of a colored image into a grey image [23]. This conversion stage into a grey image may be omitted in applications where color features are of relevance, otherwise this step is crucial because it is much simpler and faster to process an image in a grey color format [17, 117-119]. The second stage involves the denoising of a specimen image as in most cases images are not without

interference with the noise signal, which affects the visibility of the features in the specimen images [23]. The third step then includes image segmentation, which will be explained more broadly in 2.1.3. The last step involves the forming of an outline image, which can be achieved by masking the leafstalk as well as holes while keeping the outer connected region [15, 23].

Wakhare [24] proposed a similar procedure to that illustrated in Fig. 2.6 for plant-leaf feature identification applications under real-life varying lighting conditions. This procedure involves the conversion of a specimen image into grayscale, noise suppression as well as smoothing, and formation of the image outline through edge filtering. In a comparative study conducted by Ekka [25], a histogram equalization method was proven to be the most effective form of image enhancement of the grey images that were originally color images. Conversely, Kolhalkar [26] found that Red-Green-Blue (RGB) camera images offer more valuable image enhancement compared to those converted to greyscale in the context of identifying diseases on plant leaves. Therefore, this study cannot conclude which image pre-processing technique is better than the other. Rather, the application in which the image is used and the kind of image involved in that application shall be considered in the selection of an appropriate pre-processing technique.

(iii) Image Segmentation

Image segmentation is a pivotal part of image-based plant feature identification and phenotyping systems [23, 120]. Segmentation of an image involves the separation between the foreground and the background [15], that is, the isolation of the feature of interest and masking of the irrelevant part from the image [24-26]. The features of interest are normally identified by comparing adjacent pixels for similarity by looking at the three main parameters, namely, the texture, color and shape [15, 17, 121]. Table 2.1 shows a list of free data libraries available to the public for use in the image segmentation process.

Table 2-1: Summary of image processing free libraries and their supporting programming languages

A list of open libraries that supports ML-based machine plant disease classification			
Software language of implementation	*Library*	*Description*	*Open source*
R	Kern-Lab	Mechanisms for segmentation, modelling, grouping, uniqueness identification, and feature matching using kernel-based deep learning [27].	https://cran.r-project.org/
	MICE	This method can deal with data sets with missing data by computing estimates and filling in the missing data values [28].	
	e1071	Programming package containing functions for types of statistical methods, i.e., probability and statistics [29].	
	CA-RET	Offers a wide range of tools for creating forecasting analytics utilizing R's extensive model library. it contains techniques for the pre-processing learning algorithm, determining the relevance of parameters, and presenting networks [30].	

	Rweka	Data pre-processing, categorization, analysis, grouping, clustering algorithms, and image processing methods for all Java-based machine learning methods [31].	
	ROCR	A tool for assessing and displaying the accuracy of rating classifiers [32].	
	KlaR	various categorization and display functions [33].	
	Earth	Utilize the methods from Friedman's publications "Multivariate Adaptive Regression Splines" and "Fast MARS" to create a prediction model [34].	
	TREE	A library containing functions designated to work with trees [35].	
R, C	Igraph	Contains functions manipulating large graphs, and displaying [34].	
Python, R	Scikit-learn	Offers a standardized interface for putting the machine into learning algorithms practice. It comprises various auxiliary tasks like data pre-processing operations, information resampling methods, assessment criteria, and search portals for adjusting and performance optimization of methods [36].	
	NuPIC	Software for artificial intelligence that supports Hypertext Markup Language	http://numenta.org/

		(HTML) learning models purely based on the neocortex's neurobiology [37].	
Python	Caffe	Deep learning framework that prioritizes modularity, performance, and expression [38]	http://caffe.berkeleyvision.org/
	Theano	A toolkit and processor that is optimized for working with and assessing equations, particularly those using array value [39].	http://deeplearning.net/software/theano
	TensorFlow	Toolkit for quick computation of numbers in artificial intelligence and machine learning [40].	https://www.tensorflow.org/
	PyBrain	A versatile, powerful, and user-friendly machine learning library which offers algorithms that may be used for a range of machine learning tasks [41].	http://pybrain.org/
	Pylearn2	A specially created library for machine learning to make learning much easier for developers. It is quick and gives a researcher a lot of versatility [42].	http://deeplearning.net/software/pylearn2
Java	Java-ML	A collection of machine learning and data mining techniques that aim to offer a simple-to-use and extendable API. Algorithms rigorously adhere to their respective interfaces, which are maintained basic for each type of algorithm's interface [43].	http://java-ml.sourceforge.net/

ELKI	A data mining software that intends to make it possible to create and evaluate sophisticated data mining algorithms and study how they interact with database search architecture [44].	http://elki.dbs.ifi.lmu.de/
JSAT	A library designed to fill the need for a general-purpose, reasonably high-efficiency, and versatile library in the Java ecosystem that is not sufficiently satisfied by Weka and Java-ML [45].	https://github.com/EdwardRaff/JSAT
Mallet	Toolkit for information extraction, text categorization, grouping, quantitative natural language processing, as well as other deep learning uses to text [46].	http://mallet.cs.umass.edu/
Spark	Offers a variety of machine learning techniques such as grouping, categorization, extrapolation, and data aggregation, along with auxiliary features like simulation assessment and data acquisition [47].	http://spark.apache.org/
Weka	Provides instruments for categorizing, forecasting, clustering, classification techniques, and visualization of information [48].	http://www.cs.waikato.ac.nz/ml/weka/
Shark	Includes approaches for neural networks, both linear and nonlinear programming,	http://image.diku.dk/shark/

797		kernel-based learning algorithms, and other methods for machine learning [49].	
C#, C++, C	mlpack	Gives the data processing techniques as simplified control scripts, Python bindings, and C++ objects that can be used in more extensive machine learning solutions [50].	http://mlpack.org/
	LibSVM	A Support Vector Machines (SVM) library [51].	http://www.csie.ntu.edu.tcjlin/libsvm/
	Shogun	Provides a wide range of data types and techniques for deep learning issues. It utilizes SWIG to provide interfaces for Octave, Python, R, Java, Lua, Ruby, and C# [52].	http://shogun-toolbox.org/
	Multiboot	offers a quick C++ solution for enhancing methods for many classes, labels, and tasks[53].	http://www.multiboost.org/
	MLC++	Supervised machine learning methods and functions in a C++ ecosystem [52].	http://www.sgi.com/tech/mlc/source.html
	Accord	Fully C#-written machine learning platform with audio and picture analysis libraries [54].	http://accord-framework.net/

Fig. 2.7 shows a straightforward example of an image segmentation technique called threshold segmentation [55, 121-123]. Threshold segmentation is a process of converting a

color or grey scale image into a binary image with the sole purpose of making feature classification easier [55, 56]. The output binary images comprise black and white colored pixels which correspond to the background and foreground respectively, or vice-versa [26, 55, 56].

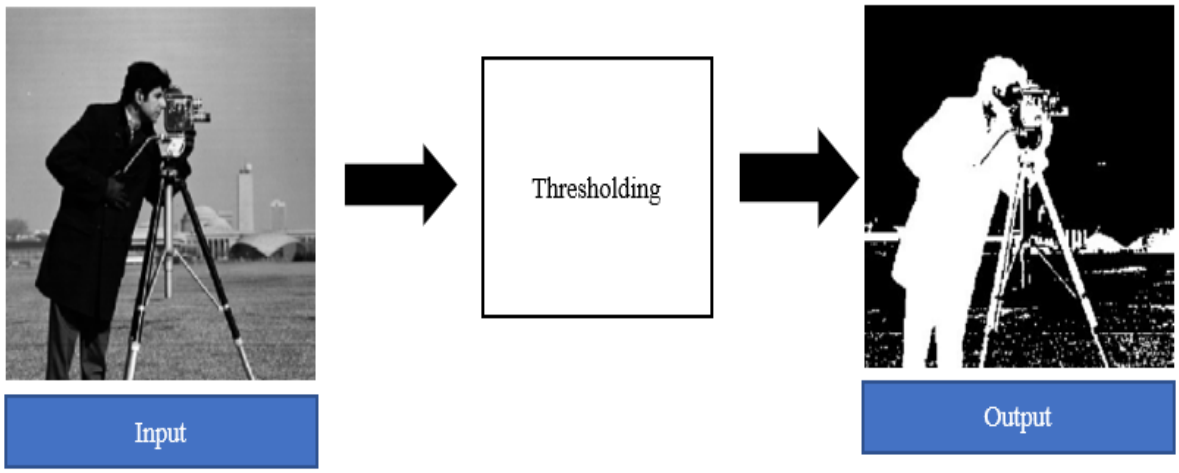


Figure 2-7: Example of thresholding image segmentation

Threshold segmentation is mathematically defined by Eq. (2.1), where *T* refers to a certain threshold intensity, *g* is the black or white pixel of a binary image, x and y denote the co-ordinate of the pixel, and *f* is the grey level of the input image [56]:

$$g(x,y) = \begin{cases} 0, if\ f(x,y) < T \\ 1\ if\ f(x,y) > T \end{cases} \quad 2\text{-}1$$

Threshold segmentation is sub-divided into three, namely global, local and adaptive thresholding [15, 57]. *Global* thresholding is applied in scenarios where there is enough distribution between the intensity distribution of the foreground compared to the background [15]. Hence a single threshold value is selected and used to distinguish between the features of significance and the background [15, 55]. *Local* thresholding is applied in cases where there is no distinct difference in intensity distribution between the background and the foreground and hence is not conducive to selecting a single threshold value [55]. In such a case, an image is partitioned into smaller images, and selects different threshold values per each partitioned picture [15]. *Adaptive* thresholding is also appropriate for images with uneven intensity

distribution where a threshold value is calculated for each pixel [57]. The *Otsu* thresholding method is another thresholding technique used for image segmentation [15]. In this technique, a measure of spread for the pixel intensity levels on either side of the threshold is listed by looping through all the reasonable threshold values [58]. The intent is to decide the threshold value where the summation of foreground and background escalates is at its minimum [15, 58]. The fundamental characteristic of the Otsu thresholding method is the fact that it implements the threshold values automatically instead of it being pre-selected by the user [58]. Eq. (2.2) is the mathematical definition for the thresholding in the Otsu method.

$$g(x,y) = \begin{cases} 1, & f(x,y) > T \\ 0, & 0\ otherwise \end{cases} \quad 2\text{-}2$$

Another segmentation method application in image processing is *watershed transformation* [59]. A grayscale image undergoes a transition called a watershed [59, 60]. In a metaphorical sense, the name alludes to a geologic catchment or drainage split that divides parallel catchments [59]. The watershed conversion locates the lines that follow the tops of ridges by treating the image it operates upon as a topographic map, with the luminosity of each pixel denoting its elevation [60]. Fig. 2.8 is an example of a watershed-segmented image where the black pixels denote the background, the grey pixels denote the features to be extracted and the white pixels correspond to watershed lines [61].

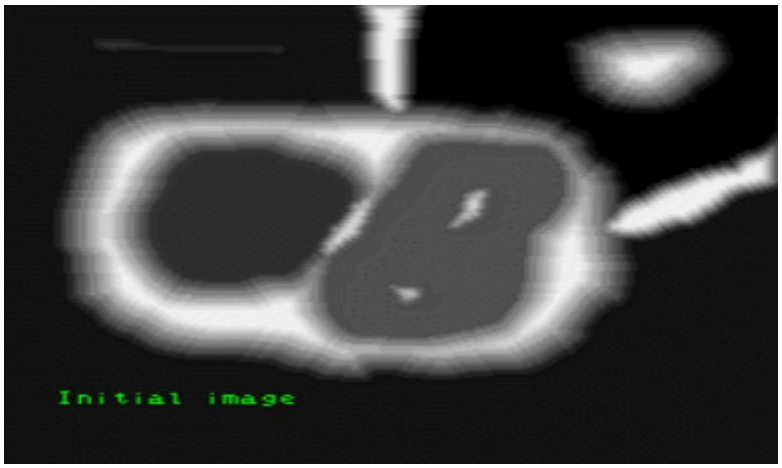


Figure 2-8: Watershed image segmentation example

On the other hand, *Grab-cut* is a very popular and innovative segmentation technique that takes into consideration the textural and boundary conditions of an image [62]. This segmentation method is based on the iterative graph-cut method where a mathematical function is derived to implement the background as well as the foreground [63]. Each pixel in an image is then assessed to decide whether it falls in the background or the foreground [62, 63]. The Grab-cut segmentation method is preferred in most applications because of minimal user interference in the operation of this technique, but it is not without its drawbacks [62]. The Grab-cut sequence cycles take a long time to implement because of the complexity of the thresholding equation [63]. The segmentation is also poor in scenarios where the background is complex and there is minimal distinction between the features of interest and the background [64]. Several distinct segmentation methods and algorithms exist in the literature. The suitability of one method is based on a particular application and hence this study is not able to rule out certain segmentation methods or merit the ones outperforming the others.

(iv) Feature Extraction

One of the foundational elements of computer vision-based image recognition is the extraction of features [65]. A feature is data that is utilized to solve a particular computer vision problem and is a constituting part of a raw image [64]. The feature vectors include the features that have been retrieved from an image [66]. An extensive range of techniques is used to identify the items in an image while creating feature vectors [62]. Edges, image pixel intensity, geometry, texture, image modifications like Fourier, Wavelet, or permutations of pixels from various color images are the primary features [46, 66]. A set of classifiers and machine learning algorithms are what feature extraction is ultimately used for [66]. The feature extraction in plant leaf disease monitoring systems is sub-divided into three spheres which include texture, color and shape [20, 21, 46, 65].

(a) Shape Features

The shape is a basic characteristic of a leaf used in the feature extraction of leaf images during image processing [66]. The primary shape parameters include the length (L), which is the displacement between the two points in the longest axis; the width (W), which denotes the displacement between the shortest axis; the diameter (D) denoting the maximum distance between the points; the area (A), which denotes the surface area of all the pixels found within the margin of a leaf picture; and the perimeter (P), which denotes the accumulative length of the pixels around the margin of a leaf picture [55, 58, 62, 64]. From the 5 defined primary characteristics of shape features, 11 distinct secondary features are formed by mathematical definitions involving two or more primary variables [59]. These 11 features are called the morphological features of a plant. The morphological features are as follows:

- *Circularity (C)* – A feature defining the degree to which a leaf conforms to a perfect circle. It is defined by Eq. (2.3) [60]:

$$C = \frac{4\pi A}{P} \tag{2-3}$$

- *Rectangularity (R)* – A feature defining the degree to which a leaf conforms to a rectangle. It is defined by Eq. (2.4) [55]:

$$R = \frac{LW}{A} \tag{2-4}$$

- *Aspect Ratio (AS)* – Ratio of width to length of a leaf. It is defined by Eq. (2.5) [55]:

$$AS = \frac{W}{L} \tag{2-5}$$

- *Smooth factor (SF)* – Ratio of leaf picture area when 5x5 and 2x2 regular smoothing filters have been used [58].
- *Perimeter to diameter ratio (PDr)* – Ratio of the perimeter to the diameter of a leaf. It is defined by Eq. (2.6) [64]:

$$PDr = \frac{P}{D} \tag{2-6}$$

- *Perimeter to length plus width ratio (PLWr)* – Ratio of the perimeter to length plus width of a leaf. It is defined by Eq. (2.7) [64]:

$$PLWr = \frac{P}{L + W} \quad \text{2-7}$$

- *Narrow factor (NFr)* – Ratio of diameter to length of a leaf [60]:
- *Area convexity (ACr)* – The area ratio between the area of a leaf and the area of its convex hull [59].
- *Perimeter convexity (ACr)* – The ratio between the perimeter of a leaf to that of its convex hull [60].
- *Eccentricity (Ar)* – The degree to which a leaf shape is a centroid [64].
- *Irregularity (Ir)* – Ratio of the diameters of an inscribed circle to the circumscribed circle on the image of a leaf [59].

(b) Colour Features

Some researchers and scholars chose to implement the colour features as the pivotal features during the extraction process [67]. The color features normally cited in the literature on leaves feature extraction include the following:

- *Color standard deviation (σ)* – A measure of how much the different colors found in an image match one another or are rather different from one another [60]. For example, an image is differentiated into an array of its basic building blocks, the pixels. Then *i* is a pointer moving across the rows of pixels in an array from the origin to the very last row *M*, while *j* is a pointer moving across the columns of pixels in an array from the origin to the very last column *N*. At any point, a pixel color intensity is defined by *p (i, j)* where *i* and *j* denote the coordinate position of a pixel in an image array. Therefore, the color standard deviation is mathematically defined by Eq. (2.8):

$$\sigma = \frac{1}{MN} \sqrt{\sum_{i=1}^{M} \sum_{j=1}^{N} p(i,j)} \quad \text{2-8}$$

- *Color mean (μ)* – A measure to identify a dominant color in a leaf image. This feature is normally used to identify the leaf type [63]. It is mathematically defined by Eq. (2.9):

$$\mu = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} p(i,j) \qquad 2\text{-}9$$

- *Color skewness (φ)* – A measure to identify color symmetry in a leaf image defined by Eq. (2.10) [21, 46]:

$$\varphi = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}[p(i,j)-\mu]^3}{MN\sigma^3} \qquad 2\text{-}10$$

- *Colour Kurtosis (Φ)* – A measure to identify a color shape dispersion in a leaf image defined by Eq. (2.11) [65]:

$$\Phi = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}[p(i,j)-\mu]^4}{MN\sigma^4} \qquad 2\text{-}11$$

(c) Texture Features

There are also several textural features referenced by authors such as Singh [68], Martsepp[69] and Ponce[70]. Using the same assumption of an image partitioned into pixels in section 2.1.4.2, the following are the textural features used for feature extraction in plant leaves:

- Entropy (Entr) – This is a measure of how complex and uniform the texture of a leaf image is, and is defined by Eq. (2.12) [68]:

$$\text{Entr} = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} p(i,j)\log_2 p(i,j) \qquad 2\text{-}12$$

- Contrast (Con) – This is a measure of how clear the features are in a leaf image, and is also referred to as the moment of inertia, defined by Eq. (2.13) [69, 70]:

$$\text{Cont} = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(i-j)^2 p(i,j) \qquad 2\text{-}13$$

- Energy (En) – This is a measure of the degree of uniformity of a grey image. It is also called the second moment, defined by Eq. (2.14) [69]:

$$\mathrm{En} = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} p^2(i,j) \qquad 2\text{-}14$$

- Correlation (Cor) – This is a measure of whether there is a similar element in a sample picture, which corresponds to the recurrence of a similar matrix within a large array of pixels. It is defined by Eq. (2.15) [68].

$$\mathrm{Cor} = \frac{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} ijp(i,j) - a_1\, a_2}{{b_1}^2 {b_2}^2} \qquad 2\text{-}15$$

Where:

$$a_1 = \sum_{i=1}^{N} ip(i,j)$$

$$a_2 = \sum_{j=1}^{N} jp(i,j)$$

$${b_1}^2 = \sum_{i=1}^{N}(i-a_1)^2 \sum_{j=1}^{N} p(i,j)$$

$${b_2}^2 = \sum_{j=1}^{N}(j-a_2)^2 \sum_{j=1}^{N} p(i,j)$$

- Difference moment inverse (DMI) – This is a measure of degree of how homogenous an image is. It is defined by Eq. (2.16) [69]:

$$\mathrm{DMI} = \sum_{i=1}^{M}\sum_{j=1}^{N} \frac{p(i,j)}{1+(i-j)^2} \qquad 2\text{-}16$$

Other textural features include the maximum probability, which is the highest response to correlation; the standard deviation and/or variance, which is the aggregate texture observed in a leaf picture; and the average illuminance, which is the average light distribution across the leaf when an image was captured, to name a few [66, 68-70]. The selection of whether to use

which color, shape or textural feature strictly depends on the application of the system being designed.

2.3.1.2. Feature Classification through Machine Learning Algorithms

The classification techniques are the machine learning algorithms used to categorize input sample data into different classes or groups of belonging or membership [3, 5, 11, 56]. These classifiers may employ supervised learning, unsupervised learning and reinforcement learning methods during their training [39]. Supervised learning occurs when a person is a trainer of the model and may use pre-formed datasets to do the training [39, 53]. Unsupervised learning occurs when there is no training data available, hence the algorithm must train itself and improve its classification efficiency by iteratively adjusting itself [5, 39, 53]. Reinforcement learning occurs when the algorithm makes classification rulings based on the feedback applied by the environment to it [12, 39]. In the case of vision-based plant disease monitoring systems, the most cited classification algorithms include Space Vector Machines (SVM), Neural Networks, k-Nearest Neighbor (k-NN) machines, and Fuzzy machines. The sub-sections below discuss these classification techniques.

(i) SVM Classifier

Support Vector Machine, sometimes known as SVM, is a predictive model used to solve both regression and classification tasks [3]. It is a supervised learning model that works well for numerous practical problems and can solve both linear and non-linear tasks [3, 71]. The SVM concept is straightforward- a vector or a hyperplane that splits the data into groups is generated by this technique [72].

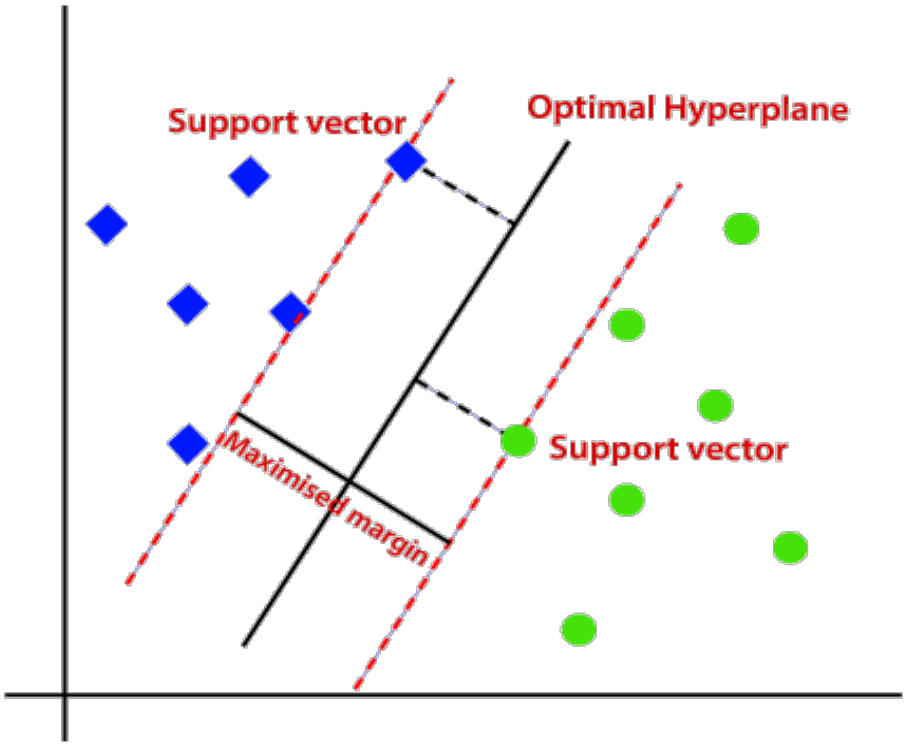


Figure 2-9: SVM classification algorithm

In Fig. 2.9, the optimal hyperplane is used to separate the two classes of data (the blue squares and green circles). The two planes (dashed lines) parallel to the optimal hyperplane are called the positive and negative imaginary planes, which are the planes passing through the closest data points to either side of an optimal hyperplane [72]. These closest points to the optimal hyperplane are called the support vectors and are used to determine the exact position of an optimal hyperplane [73]. There might be several possible hyperplanes, but the optimal hyperplane is the one with the maximum marginal distance, which is the distance between the two marginal planes [72, 73]. The maximized margin results in a more generalized solution compared to smaller margins and should the training data change, the algorithm with a smaller margin will have accuracy challenges [73]. In some cases, data classes are not always easily separable with a straight light or place, as in the case of Fig. 2.9. Therefore, when data classes show a property of non-linearity, transforming a space in which these data classes occur from a low dimension (often 2-dimensional) into a high-dimension (often 3-dimensional) space using the Kernel method. The Kernel method is a computation of a dot product of the dimensions in

the new high-dimension space [72-74]. Eq. (2.18) gives the general solution of a hyperplane where $\vec{x}$ is any data point or support vector, $\vec{\omega}$ is the weight vector that applies the bias of the support vectors and ω_0 is the constant [74].

$$g(\vec{x}) = \vec{\omega}\vec{x} + \omega_0 \qquad 2\text{-}17$$

$$where\ g(\vec{x}) = \begin{cases} 1, & for\ Class\ 1\ vectors \\ -1, & f - or\ Class\ 2\ vectors \end{cases}$$

(ii) ANN Classifier

An ANN is a supervised learning model that is a collection of interlinked input and output nodes in which each link has an associated bias value called a weight [75]. A single input layer; one or perhaps more intermediate layers which are normally called hidden layers; and one or more output layers make up the structure of an ANN [75, 76]. The weight of each connection is modulated as the network operates to facilitate neural network learning [76]. The performance of the network is enhanced by adjusting the weight continuously [75]. ANN can be divided into two groups based on connection types: feed-forward networks and recurrent networks [33]. In contrast to recurrent neural networks, feed-forward neural networks do not have cycle-forming connections between units [76]. The architecture, transfer function and learning rule all have an impact on how a neural network behaves[49, 76]. The weighted total of input triggers the activation of neural network neurons [75]. Fig. 2.10 shows a generalized model of an ANN model with the input layer, the hidden intermediate layer (purple layer), and the output layer.

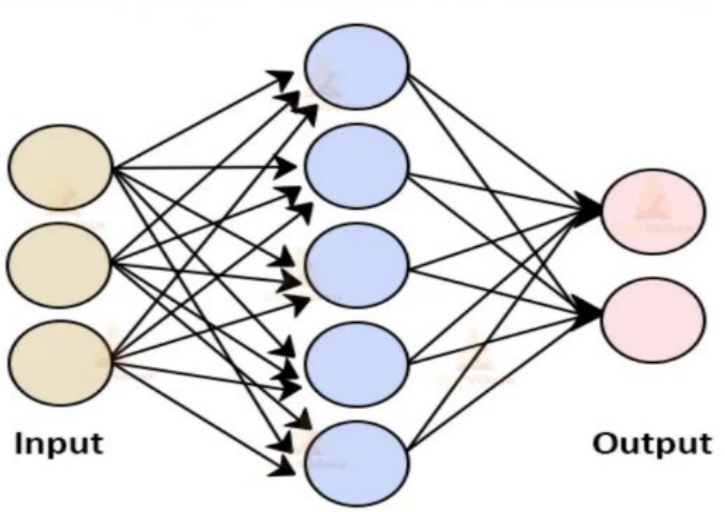


Figure 2-10: A simplified ANN model architecture

(iii) k-NN Classifier

The k Nearest Neighbors algorithm, sometimes known as kNN, is the most straightforward machine learning technique [78]. It is a non-parametric technique used for problems involving regression and classification [74, 78]. Non-parametric implies that no dataset for initial training is necessary [78]. Therefore, kNN does not require the use of any presumptions [79]. The k-closest training examples in the feature space provide the input for classification and regression tasks respectively [78]. Whether kNN is applied for classification or regression determines the results [79]. The outcome of the kNN classifier is a class of belonging [74, 78, 79]. Based on the predominant kind of its neighborhood, the given data point is classed [79]. The input point is awarded to the category that has the highest frequency amongst its k closest neighbors [78]. In most cases, k is a little positive integer such as 1. The result of a kNN regression is just a value of the property for the attribute. The aggregate of the variables of the k closest neighbors constitutes this number [79].

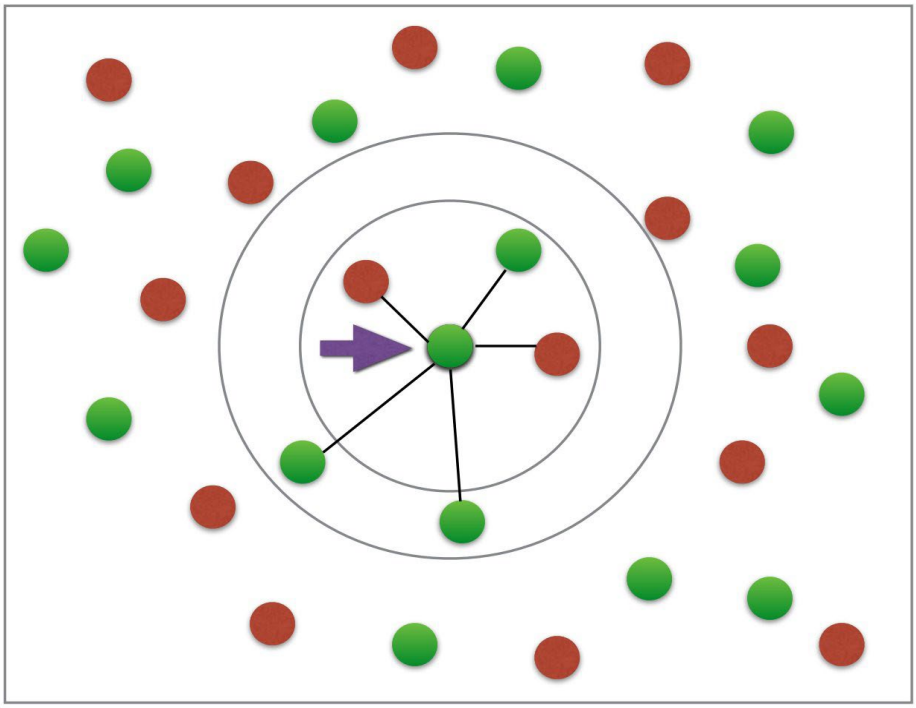


Figure 2-11: Classification principle of a kNN model

Fig. 2.11 shows a space with numerous data points or vectors that can be classified into two classes: the red class and the green class. Now, assume there exists a data point at any location in the space shown in Fig. 2.11 that is unknown as to whether it belongs to either the red or green class. The kNN will then go through the following computational steps to assign that point a class of belonging:

- Take the uncategorized data point as input to a model,
- Measure the spatial distance between this unclassified point to all the other already classified points. The distance can be computed via Euclidean, Makowski, or Manhattan formulae [80],
- Check the points with the shortest displacement from the unknown data point to be classified for a certain K value (K is defined by the supervisor of the algorithm) and separate these points by class of belonging [80], and
- Select the correct class of membership as the one with the most frequent vectors as the neighbors of the unknown data point [80].

The most cited method of computing the spatial distance between the data point p to be classified and its neighbors q_n is the Euclidean formulae defined in Eq. (2.18) [74, 80]:

$$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$
$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \quad \text{2-18}$$

(iv) FUZZY Classifier

The fuzzy classifier system is a supervised learning model that enables computational variables, outputs and inputs to assume a spectrum of values over predetermined bands [81]. By developing fuzzy rules that connect the values of the input variables to internal or output variables, the fuzzy classifier system is trained [82]. It has mechanisms for credit assignment and conflict resolution that combine elements of typical fuzzy classifier systems [81]. A genetic algorithm is used by the fuzzy classifier system to develop suitable fuzzy rules [83].

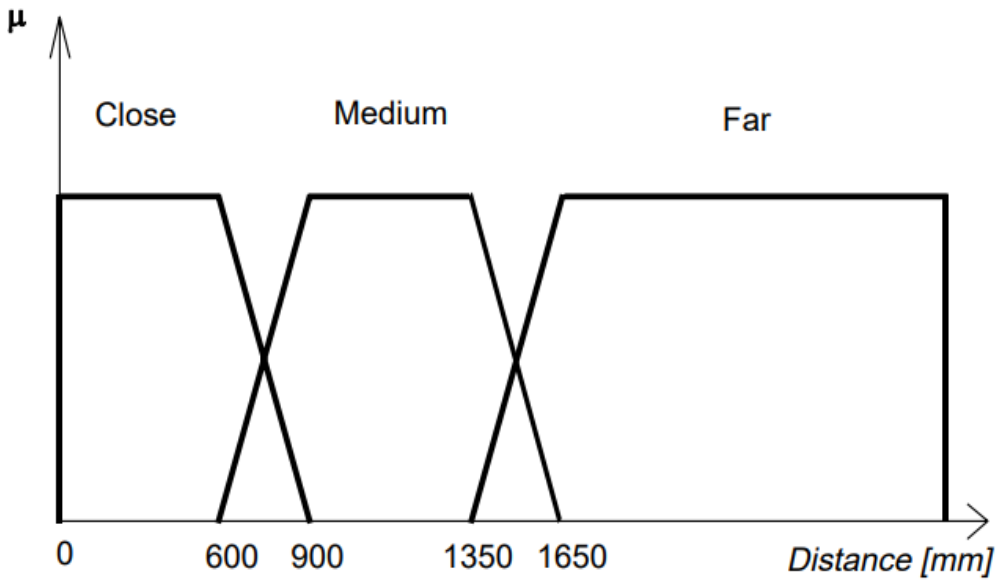


Figure 2-12: Example of FUZZY sets for classification

As shown in Fig. 2.12, fuzzy sets display a continuous membership, and a data point membership classification can be ruled as the extent (μ) to which it belongs to a certain fuzzy set. For example, 690 mm in Figure 2.12 has a degree of membership μ (960) on the close fuzzy set that is 0.7. It can also be seen from Fig. 2.12 that a data point can belong to multiple fuzzy sets, and the degrees of membership to each set may or may not (in the intersection points)

differ, since some fuzzy sets overlap with each other. Table 2.2 summarizes the advantages and disadvantages of all the classification techniques discussed in this section.

Table 2-2: PRO & CONS of different classification methods

Pros & Cons of Different Classification Methods Most Used in Plant Phenomics and Disease Monitoring	
1. Support Vector Machine (SVM)	
Advantages	**Disadvantages**
• Works very accurately when there is a clear formation of a hyperplane [74].	• Accuracy difficulties with a large amount of training data [71].
• Works more accurately on high-dimension spaces like 3-D and 4-D [51].	• Susceptibility to noise and overlapping data classes [75].
• Saves memory space [71].	• The number of characteristics for a single data set must not exceed the number of data points in the training set [74].
2. Artificial Neural Network (ANN)	
• Capable of multitasking [76].	• Complex programming algorithms [75].
• The machine is learning continuously, and the accuracy is improving iterative[50].	• Accuracy is data dependent; more training data translate to more accurate classification and the opposite is true [75].

• Have many applications (e.g., mining, agriculture, medicine, and engineering) [59].	• Hardware reliance (cost, complexity, and maintenance) [33].
3. k-Nearest Neighbour (k-NN)	
• No initial training period [74].	• Accuracy difficulties with a large amount of training data [79].
• Simple to add new data to the model to extend its scope [80].	• Not suitable for high-dimensional space [80].
• Relatively easy to implement with only the two parameters to work out, the k value and the geometric distance between the points [78].	• Susceptibility to noise and outliers [74].
4. Fuzzy classifier	
• Unclear, distorted, degraded, or vague input data is accommodated by the model [81].	• Depending on people's experience and expertise [82].
• More flexibility and ease to change the rules [83].	• Require excessive supervision in the form of testing and validation [82].

• Robust in applications with no exact input format [82].	• The is no universal approach to implementing fuzzy classification models which adds to their inaccuracy [83].

2.3.2. Summarized Literature Survey: Plant Disease/Nutrient Deficiency Monitoring Systems

Many authors in the literature have proposed plant disease/pest/weed detection systems that employ the above-described general format. Table 2.3 summarizes the key research publications that were consulted during this research study on these systems.

Table 2-3: Summary of the literature survey on the plant disease/pest/weed detection systems

Summary of publications on plant disease/pest/weed detection systems by separate authors					
Classification Method	*Plant/Crop*	*Reference*	*Number of diseases*	*Disease*	*Results*
	Maize	[84]	1	Not Specified	79% accuracy
	Grapefruit, Lemon, lime	[85]	2	Canker And Anthracnose Diseases	95% accuracy for both
	Grape	[86]	2	Downy Mildew And Powdery Mildew	88.89% accuracy for both

	Oil palm	[3]	2	Chimaera and Anthracnose	97% and 95% accuracy respectively
	Potato	[87]	4	Late Blight and Early Blight	95% for both
	Grape	[10]	3	Black Rot, Esca, And Leaf Blight	Not specified
	Tea	[88]	3	Not Specified	90% accuracy
SVM Classification	Soybean	[85]	3	Downy Mildew, Frog Eye, And Septories Leaf	90% accuracy average
	Tomato	[89]	6	Not Specified	96% accuracy
	Rice	[90]	Not specified	Pests Diseases	92% accuracy
	Soybean	[91]	1	Charcoal Rot	90% accuracy
	Cucumber	[92]	1	Downy Mildew	Not specified
	Rice	[93]	1	Rice Blast	93% accuracy
	Rice	[94]	1	Rice Blight	80% accuracy
	Tea	[95]	1	Not Specified	90% accuracy
	zucchini	[96]	1	Soft-Rot	Not specified

ANN Classification	Not specified	[97]	4	Alternaria Alternata, Anthracnose, Bacterial Blight, Cercospora Leaf Spot	96% accuracy average
	Grapefruit	[98]	3	Grape-Black Rot, Powdery Mildew, Downy Mildew	94% accuracy average
	Apple	[99]	3	Apple Scab, Apple Rot, Apple Blotch	81% accuracy average
	Pomegranate	[100]	3	Bacterial Blight, Aspergillus Fruit Rot, Gray Mold	99% accuracy average
	Not specified	[101]	4	Early Scorch, Cottony Mould, Late Scorch, Tiny Whiteness	93% accuracy average

Cucumber	[102]	2	Downy Mildew, Powdery Mildew	99% accuracy average
Pomegranate	[103]	4	Leaf Spot, Bacterial Blight, Fruit Spot, Fruit Rot	90% accuracy average
Groundnut	[104]	1	Cercospora	97% accuracy
Pomegranate	[105]	1	Not Specified	90% accuracy
Cucumber	[106]	1	Downy Mildew	80% accuracy
Rice	[107]	3	Bacterial Leaf Blight, Brown Spot, Leaf Smut	96% accuracy average
Citrus	[108]	5	Anthracnose, Black Spot, Canker, Citrus Scab, Melanose	90% accuracy average
Wheat	[109]	4	Powdery Mildew, Rust Puccinia	Not specified

				Triticina, Leaf Blight, Puccinia Striifomus	
k-NN Classification	Not specified	[110]	5	(YS) The Yellow Spotted, (WS) White Spotted, (RS) Red Spotted, (N) Normal, and (D) Discolored Spotted	86% Accuracy
	Groundnut	[78]	5	Early Leaf Spot, Late Leaf Spot, Rust, Early and Late Spot Bud Necrosis	96% Accuracy
	Tomato, Corn, Potato	[111]	Not Specified	No Disease: Leaf Recognition	94% Accuracy (Corn) 86% Accuracy (Potato) 80% Accuracy

	Tomato	[112]	3	Rust, Early and Late Spot Bud Necrosis	95% Accuracy
	Banana	[113]	2	Bunchy Top, Sigatoka	99% Accuracy
	Tomato	[114]	3	Rust, Early and Late Spot Bud Necrosis	97% Accuracy
	Rice		4	Bacterial Blight Of Rice, Rice Blast Disease, Rice Tungro, False Smut	88% Accuracy Average
Fuzzy Classification	Mango	[83]	3	Powdery Mildew, Phoma Blight, Bacterial Canker	90% Accuracy Average
	strawberry	[115]	1	Iron Deficiency	97% Accuracy

			Bacterial Blight, Leaf Curl, Root Rot, Verticillium Wilt, Anthracnose, Seed Rot, Tobacco Streak Virus, Tropical	
Cotton, Wheat	[116]	18	Rust, Fusarium Wilt, Black Stem Rust, Leaf Rust, Stripe Rust, Loose Smut, Flag Smut, Complete Bunt, Partial Bunt, Earcockle, Tundo	99% Accuracy Average
Soybean	[19]	1	Foliar	96% Accuracy

Cotton	3	Bacteria Blight, Foliar, Alternaria	95% Accuracy Average

2.4. Opportunities Identified

During survey of the available literature presented in Section 2.3, the opportunities which the author of this thesis believes have seen little or no interest from the researchers are as follows:

- Little or no literature discussed the real-time monitoring of the onset signs of diseases before they spread throughout the whole plant part.
- Few publications discussed real-time monitoring and real-time mitigation measures such as actuation operations, spraying pesticides and spraying fertilizers, to name a few examples.
- Very little research discusses the combination of these monitoring and phenotyping tasks into one system to reduce costs and improve technology availability to farmers and add convenience.
- Most studies utilize single image sensors to capture the input images to be processed by a classification model. This may be problematic, especially in cases where spherical or cylindrical specimens are used for classification as some parts may be hidden from the camera's line of sight.
- Few studies focus on the post-harvesting benefits of these classification models.
- No study was encountered during the literature review discussing the relationship between the classification efficiency and accuracy of these classification models to the lighting conditions in which they operate.
- Few studies discuss the role that can potentially be played by these plant disease classification models in high throughput and high-speed production plants.

- No study was encountered that discusses the estimation of the disease stage of infection, e.g., early stage or full infection stage.
- The image pre-processing techniques that have been proposed on the reviewed literature are based either on the RGB or greyscale images. Individual Red, Green or Blue colour plane images seldom provides clear details on images and a precise distinction between the foreground and the background compared to both the RGB and greyscale images.

2.5. Conclusion

This thesis has presented the background to the research in precision agriculture. The most pressing challenge in precision agriculture was identified as the monitoring of disease/pest/weed/nutrient deficiency in crops, backed by the literature trends. The basing principles of a disease/pest/weed/nutrient deficiency monitoring system were discussed. This covered the image processing steps, namely image acquisition, image pre-processing, image segmentation and feature extraction, as well as the different classification algorithms most cited in literature, namely the SVM, the ANN, k-NN and Fuzzy classifiers. The summary of a literature survey regarding the disease/pest/weed/nutrient deficiency monitoring systems was presented. The main opportunities observed during a literature survey were then presented.

The literature survey revealed that this field of precision agriculture is a relatively new research field and that considerable research focus and hence progress has been observed over the past two decades. It is also apparent that there still exists an array of opportunities that still require to be filled with further research, such as those presented in Section 2.2.2. Much more can still be done to further improve the accuracy levels of some monitoring systems presented in Table 2.3, such as increasing the amount of training data.

CHAPTER 3: METHODOLOGY

3.1. Introduction

This chapter outlines the research methodology employed throughout this research study. The methodology includes the assumptions made at the beginning of this research, the main research methods, data collection and analysis, materials and equipment used, difficulties encountered, and the overall research evaluation and conclusion. The coming sub-sections each address a specific aspect of methodology for this research study.

3.2. Outline of the research methods

This research project is greatly a quantitative study because it generates directly quantifiable results. These results include the training and validation accuracy and efficiency, the proposed model sorting efficiency and accuracy, as well as the ability of the proposed system to identify each fruit's categorical class with a certain level of confidence interval. The general research methodology utilized is as follows:

- Extensive literature survey: Several machine learning and deep learning-based plant disease monitoring open access publications have been consulted to gauge the status of research in this field and to identify a research niche. During this survey, several opportunities were identified in this research field, and a few are listed below:
  - ❖ Little literature discussed the real-time monitoring of the onset signs of diseases before they spread throughout the whole plant part.
  - ❖ Few publications discussed real-time monitoring and real-time mitigation measures such as actuation operations (such as sorting), spraying pesticides and spraying fertilizers, to name a few examples.
  - ❖ Very little research discusses the combination of these monitoring, phenotyping and actuation operations into one system to reduce costs and improve technology availability to farmers and add convenience.

The research study takes advantage of the opportunities identified by proposing several improvements to the classical plant disease detection models. One of the proposals is a model that performs the actuation operation after classification, and the same system combines different functionalities that are commonly known as stand-alone systems into one universal

system. Hence, disease detection, quality assurance, fruit grading and overall sorting are possible with a singular system.

- The next step is the design of a sample classical disease detection system. This step is featured to first validate the research opportunities identified in the literature, to learn how different critical performance parameters such as training dataset amount versus classification accuracy, and training dataset variability versus classification accuracy relate; and lastly, to serve as a point of reference for evaluating the later proposed system. In the case of this research study, a healthy-black rot-affected orange classification model was designed.

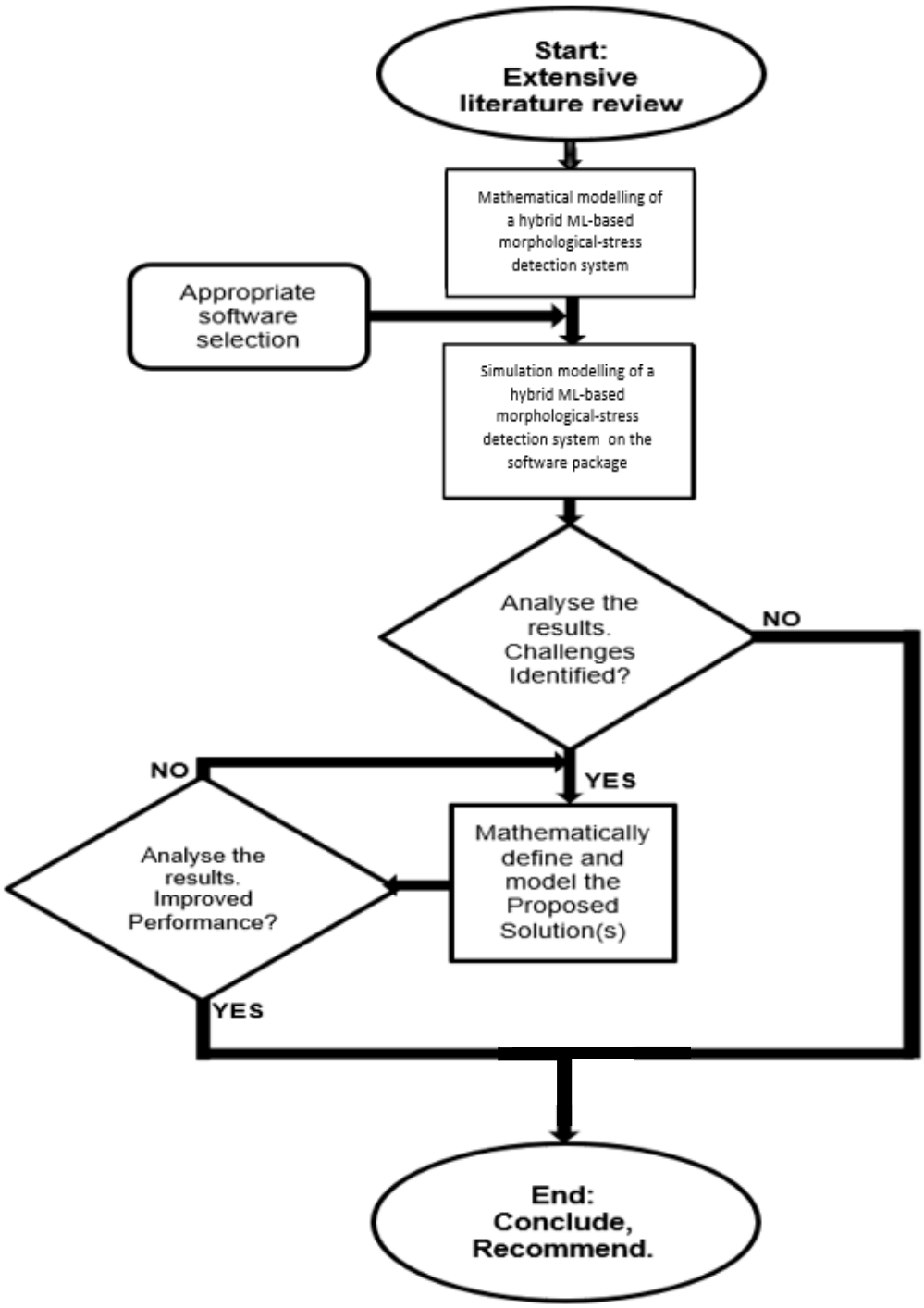


Figure 3-1: Research methodology flow diagram for designing a proposed system

- Then, the proposed systems are conceptualized, designed, implemented and hence tested against the operation of a classical system designed in the previous step. One of the proposed systems is called a Hybrid Fruit Disease-Quality Monitoring and Sorting Model (H-DQMSM) and this system is based on Artificial Intelligent Machine Learning, particularly the deep learning branch. The training method that was used to train this deep

learning model is called transfer learning, which is training a previously existing deep learning model to perform the new tasks stipulated by the designer. A model used is GoogleNet, which has the original capacity to classify 1000 different objects and it was retrained to classify 6 different categories. This model was selected to allow for future expansion plans. Therefore, more classes or capabilities can be added to a proposed system in the future.

- The last step is evaluating a Hybrid Fruit Disease-Quality Monitoring and Sorting Model against a classical model and concluding on the results. Fig. 3.1 shows the flow diagram representation of the outlined methodology process followed during this design phase of the Hybrid-DQMS Model.

3.3. Data collection and analysis

Data collection occurred in three instances during this research study. It occurred during the collection of machine learning and deep learning disease detection system publications for a literature survey; during the collection of training, validation and testing image datasets; and lastly during the comparative analysis of the proposed systems' results relative to those of a classical system. The related publications consulted were on a year-to-year basis over the last two decades. The training and validation dataset collection was achieved through the open-source dataset library called Kaggle, while the testing dataset was mainly acquired through capturing snapshot images from a live-feed video of the actual fruit samples utilizing a Macally webcam, as shown in Fig. 4.4 (in Chapter 4). The operational data analysis and evaluation of a Hybrid Fruit Disease-Quality Monitoring and Sorting Model were obtained through practical testing of this system, as shown in Chapter 5.

3.4. Materials and equipment

The list of materials and equipment used during this research study is as listed:

- MATLAB R2022a modeling software package (deep learning toolbox);
- Host computer to serve as a microprocessor holding the classification models and executing the program instructions;
- A Macally UVC webcam;
- An open-source image dataset library; and

- Sample fruits to create a testing image dataset collected from the local vending market (practical dataset).

3.5. Difficulties and limitations encountered

Several difficulties were encountered during the undertaking of this research study, as listed below:

- The scarcity of image datasets. For example, this study originally aimed to include the classification of punched fruit (apples and oranges) under the quality assurance checks. However, due to the unavailability of sufficient image data, one type of quality defect was selected to represent all related defects, including punchers.
- Creating one's own dataset is tedious and extremely time-consuming, either by hunting for images from the internet or by taking new images from the actual fruit samples. If one opts to take images from the physical samples, it proves difficult to obtain enough physical samples for a particular categorical class.
- Supervised machine learning requires an expert's input at the beginning of the training process to correctly classify different categories such as black rot and botch disease in oranges and apples. Hence, in theory, when the author of this thesis generated his dataset, he may have had errors since he is not an expert in fruit biology, which may compromise the accuracy of the proposed classification model. However, careful considerations have been made to minimize such an occurrence.
- The available datasets do not present enough variance amongst images of a dataset. For example, a few apples may be harvested with a leaf (Fig. 3.2 (a)) while the rest may not (Fig. 3.2(b)). In such cases, for more accurate results, both these variations should be strongly represented in the training and validation dataset. Otherwise, the testing phase will be less accurate.

(a) (b)

Figure 3-2.: (a) Apple harvested with a leaf; (b) Apple harvested without a leaf

3.6. Overall research evaluation

This section covers a brief research evaluation, whilst the main evaluation is covered in Chapter 5. The conceptual evaluation in this section will be done against the objectives of the research study. Four key proposals have been made in this study, namely:

- An Improved Image Pre-processing Procedure for a Fruit Disease Detection Technique;
- A 3-Dimensional Image Input-Based Fruit Disease Detection Model Using Deep Learning Technology;
- A Multi-Camera Image Input-Based Plant Disease Classification Model; and
- A Hybrid Fruit Disease-Quality Monitoring and Sorting Model (H-DQMS Model).

Therefore, Table 3.1 below summarily evaluates each of these 4 proposals against the objectives of this research study.

Table 3-1: Evaluation of a Hybrid Fruit Disease-Quality Monitoring and Sorting Model

Evaluation of the Proposed Models Against the Study's Objectives	
An Improved Image Pre-processing Procedure for a Fruit Disease Detection Technique	
Objective	*Evaluation*
• Improving the efficiency of the disease classification operation	This technique helps to improve the color intensity contrast between the background and

	foreground. A better contrast implies a clear-cut segmentation of the important feature which are the disease symptoms or general defects on the fruit surface in the case of this study.
• Improving the accuracy of the disease classification operation	Once the classification efficiency is improved because of a clear-cut feature segmentation, The classification accuracy of the models utilizing this proposed model will be improved.
• Maximizing a fruit (such as oranges or/and apples) farm yield by minimizing losses during the harvesting operations.	An improved classification efficiency and accuracy translate to more efficient detection of crop diseases on a farm which then informs corrective actions such as the use of fertilizers. This helps minimize the yield losses at the end of a farming cycle.
A 3-Dimensional Image Input-Based Fruit Disease Detection Model Using Deep Learning Technology	
• Improving the efficiency of the disease classification operation	Classification efficiency is improved by eliminating 'blind spots' and enabling a classification model to have a 3-dimensional view of spherical and/or cylindrical samples such as fruits, stems, seeds, etc.
• Improving the accuracy of the disease classification operation	The classification accuracy of this model is greatly improved even for plant samples where a disease is at the onset stage or for plant samples where the disease symptoms are not distributed

	evenly across the surface area of a spherical/cylindrical plant sample.
• Maximizing a fruit (such as oranges or/and apples) farm yield by minimizing losses during the harvesting operations.	An improved classification efficiency and accuracy translate to more efficient detection of crop diseases on a farm which then informs corrective actions such as the use of fertilizers. This helps minimize the yield losses at the end of a farming cycle. Moreover, this model can be applied in the sorting, grading, and quality checks operations.
A Multi-Camera Image Input-Based Plant Disease Classification Model	
• Improving the efficiency of the disease classification operation	Classification efficiency is improved by eliminating 'blind spots' and enabling a classification model to have a 3-dimensional view of spherical and/or cylindrical samples such as fruits, stems, seeds, etc.
• Improving the accuracy of the disease classification operation	The classification accuracy of this model is greatly improved even for plant samples where a disease is at the onset stage or for plant samples where the disease symptoms are not distributed evenly across the surface area of a spherical/cylindrical plant sample
• Maximizing a fruit (such as oranges or/and apples) farm yield	An improved classification efficiency and accuracy translate to more efficient detection of crop diseases on a farm which then informs

by minimizing losses during the harvesting operations.	corrective actions such as the use of fertilizers. This helps minimize the yield losses at the end of a farming cycle. Moreover, this model can be applied in the sorting, grading, and quality checks operations.
A Hybrid Fruit Disease-Quality Monitoring and Sorting Model (H-DQMS Model)	
• Maximizing a fruit (such as oranges or/and apples) farm yield by minimizing losses during the harvesting operations.	The proposed system achieves this objective by reduction of fruit sorting turn-around time and by minimizing the losses incurred by human error in the manual sorting operation.
• Inform timely corrective actions by quickly identifying diseases on fruits.	The proposed system achieves this objective but identifying fruit disease and informing timely corrective actions such as the deployment of a certain fertilizer. Hence, the proposed system operation is not limited to the harvesting season only but can also be utilized during the growing phase of fruits
• Improve a farm's revenues.	This is achieved through time-saving and error elimination.
• Improve and maintain good supplier-customer relations.	This is achieved by ensuring that only the top-quality fruit is dispatched to customers as per their expectations, hence drastically reducing disputes.
• Economic viability to developing farmers	This is achieved by integrating several capabilities that are originally stand-alone

	systems (classical model) into a single universal system (proposed system) hence drastically reducing the capital expenditure (CAPEX) require for commissioning the proposed system.

3.7. Conclusion

This chapter outlined the overall methodology of this research study. The objectives of this study were fulfilled as per the brief evaluation presented in Section 3.6, as well as a more detailed evaluation in Chapter 5. The full details of future work studies will be detailed in Chapter 6. Chapter 4 utilizes the tools outlined in Chapter 2 to design a classical classification model called a Healthy-Black Rot Orange Classification model. This model classifies black rot-affected oranges from healthy oranges and its purpose in this study is to serve as a frame of reference for evaluating the models to be proposed in Chapter 5.

CHAPTER 4: MODELLING AND TESTING OF A FRUIT DISEASE DETECTION MODEL

4.1. Introduction

The purpose of this chapter is to conceptually design and implement a classical plant disease detection model and test it with orange fruits to validate its functionality. The tools outlined in Chapter 2 are now put into action in this chapter and the results will serve as a reference for a comparative analysis of the improved models to be proposed in Chapter 5. The implementation of this model will take place in the MATLAB environment. The model design will be undertaken in two major parts, namely the image processing, displayed by a block diagram in Fig. 4.1, and the classification of a disease which will be implemented via a CNN. A CNN was selected because it appears most frequently amongst the machine learning algorithms utilized for disease detection in plants and because it has a built-in feature extraction capability.

Figure 4-1: Block diagram of the image processing design sequence for the plant disease system detection

4.2. Image Processing Model Design

As discussed in Chapter 2, image processing is a crucial step in ML-based disease/pest/weed detection systems where raw image properties are altered in a manner that aids the extraction of useful features for classification purposes. Image processing is an example of a signal processing operation where the input and output are a raw image and a feature-enhanced image respectively [3]. Image processing ranges from simple operations such as cropping, clipping, re-sizing, filtering, and de-blurring to complex operations such as segmentation and feature extraction. Several image-processing techniques for ML-based disease/pest/weed detection systems have been proposed in the literature. This sub-section develops an image pre-processing technique that improves the contrast between the background and foreground in a fruit image for application in a disease detection model.

4.2.1. Image Acquisition

A charge-coupled device (CCD) and a complementary metal oxide semiconductor (CMOS) image sensors are the popular choices for capturing images of plant leaves, stems, flowers or fruits for disease, pest or weed classification and are the most cited devices in literature. At the time of the compilation of this work, a CMOS camera had been ordered and the estimated time of arrival (ETA) was 96 business days. Fig. 4.2 shows a Camera Serial Interface (CSI) – 2 Raspberry Pi camera module capable of capturing images and live-stream videos. This camera module consists of two major parts, an image sensor and an image interface.

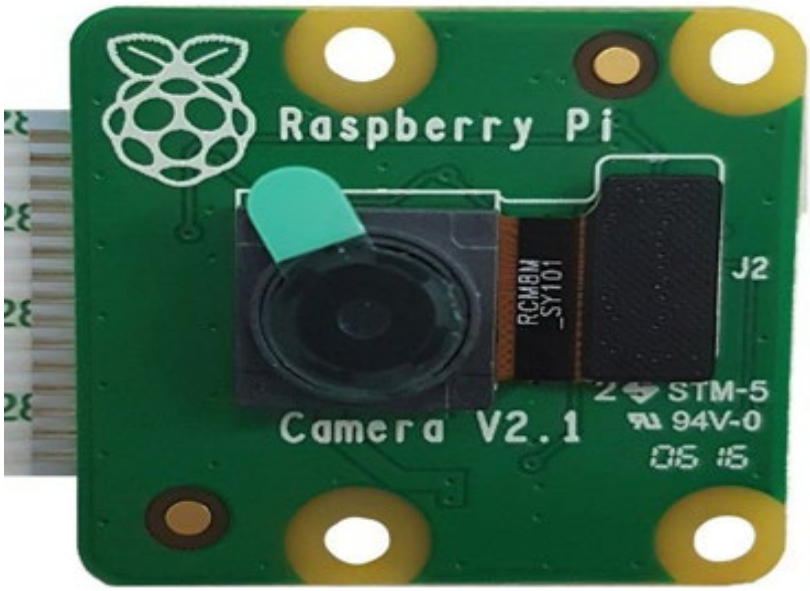


Figure 4-2: CSI-2 Raspberry Pi Camera Module

The image sensor consists of a 2x2 matrix of photodetectors that are sensitive to the primary colors Red-Green-Blue (RGB). Therefore, an image sensor converts an analog image into a digital image by mobilizing the photodetectors to assign a light intensity value between a range of [0-255] for each of the RGB primary colors, per every pixel in an image. The result of an image sensor is a 3-depth 2x2 RGB matrix holding the color intensity values of the R, G and B planes (a digital image) for corresponding pixels in an analog image such as Fig. 4.3. The image interface, more specifically the mobile industry processor interface (MIPI) Alliance CSI-2 interface, then transfers a digital image to a Rasberry Pi computer for further processing. A CSI-2 Raspberry Pi Camera Module uses a CMOS image specifically called OmniVision 5647 because it is less expensive compared to a CCD sensor and can counter the light blooming effect. However, it is susceptible (to some extent) to image distortion resulting from the rolling

shutter effect. Therefore, the relative speed between the camera and the fruits/leaves/roots/pests/weed being captured by the camera should be managed to dampen the image distortion. A CSI-2 Raspberry Pi Camera Module is rated at 8-megapixel and has a 3280 x 2464 still image resolution. The maximum image data transfer rate at full high-definition (HD) resolution of 1920 pixels columns and 1080 pixels rows, simply abbreviated as 1080p, is 30 frames-per-second (fps). At HD resolution of 1280 pixels columns and 720 pixels rows, simply abbreviated as 720p, the maximum image data transfer is 60 fps. A CSI-2 Raspberry Pi Camera Module has length-width-depth dimensions of 23.86 x 25 x 9 mm and weighs 3 grams.

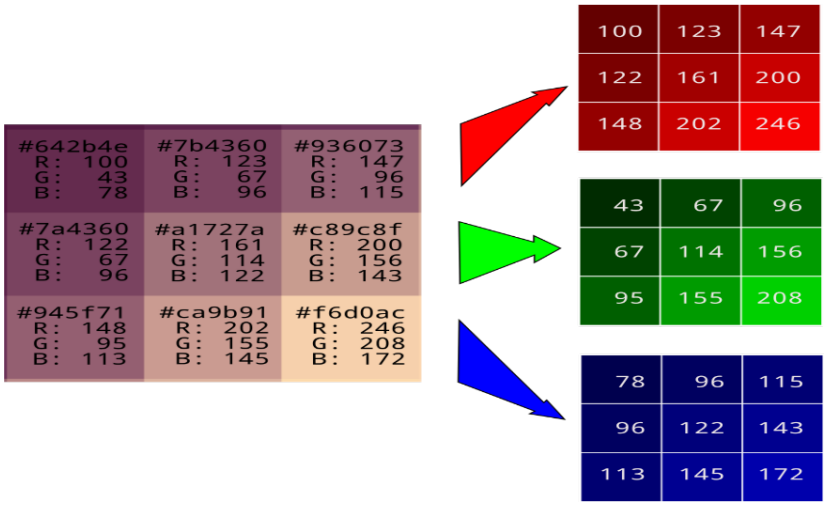


Figure 4-3: An example R-G-B color intensity 3-depth 2x2 matrix

Figure 4-4: A UVC webcam

However, the images used in the results presented by this thesis were captured using a USB video class (UVC) webcam (refer to Fig. 4.4) while waiting for a CSI-2 Raspberry Pi Camera Module and a Rasberry Pi kit on which to deploy a disease/pest/weed detection model. This webcam has a rated resolution and image transfer speed of 1080p and 30 fps respectively.

4.2.2. Image Pre-processing

Several different image pre-processing methods have been proposed by scholars in the literature. Wang [10] proposed greyscaling, image filtering, image binarization and edge detection as effective image pre-processing steps for plant disease detection systems. Wakhare [1] also proposed a similar image pre-processing procedure to that discussed by Wang [10]. Ekka [12] proposed a histogram equalization technique as a crucial step of image pre-processing through contrast enhancement, which lays a good foundation for subsequent image segmentation and hence feature extraction. On the other hand, Kolhalkar [12] discovered that RGB images occasionally provide better image segmentation compared to greyscale images and hence it is not mandatory to convert RGB images to grayscale images during pre-processing. This goes to prove that there is no single pre-processing technique that is superior to the other. The suitability of one pre-processing technique relies on the nature of sample images and the target image enhancement to be achieved. The proposed pre-processing procedure for this study was implemented using MATLAB R2022a software and is as follows:

4.1.2.1. RGB image acquisition

Fig. 4.5 shows 3 sample pictures of an orange fruit scientifically known as Citrus × Sinensis from the Rutaceae family. Fig. 4.5(a) shows a healthy orange which serves as a control in this pre-processing procedure. Fig. 4.5(b) shows an orange affected by a disease called Citrus Black Rot. This disease is caused by a fungal infection and can be distinguished by dark spots of decaying tissue on the surface of an orange. This disease induces early fruit drops and incurs serious yield losses at the end of a farming cycle. The disease is air-borne and water-borne, and it spreads when spores are released in rainwater and wind during storms. It also spreads through contact with contaminated leaves, stems, fruits or even debris carried by stormwater. Fig. 4.5(c) shows an orange with Bug Pests feeding on it, which causes widespread destruction of the production and affects the end yield. Fig. 4.5(a) is 500x500 pixels wide and was captured by a Macally UVC webcam, as shown in Fig. 4.4. Fig. 4.5(b) is 545x525 pixels wide while Fig. 4.5(c) is 300x226 pixels wide, and both were acquired from Google.com open libraries.

(a) (b) (c)

Figure 4-5: (a) Healthy orange RGB image; (b) Black-rot affected orange RGB image ; (c) Pest-infested orange greyscale RGB image

4.1.2.2. Image Re-sizing

The second step of this pre-processing procedure is re-sizing images to one standard size, and that size should match the input size of an ANN that will be utilized to perform a disease classification. Hence, 226x226 pixels will be regarded as a standard image size throughout this thesis. The following piece of code exports the images in Fig. 4.5 into MATLAB's workspace and performs the re-sizing operation into the standard size. The re-sized 226x226 RGB images stored in MATLAB's workspace are shown in Fig. 4.6.

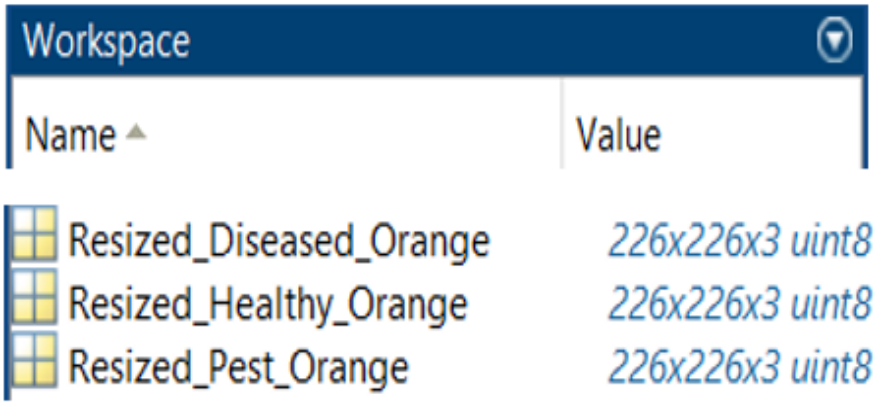


Figure 4-6: Re-sized 226x226 RGB images stored in MATLAB's workspace

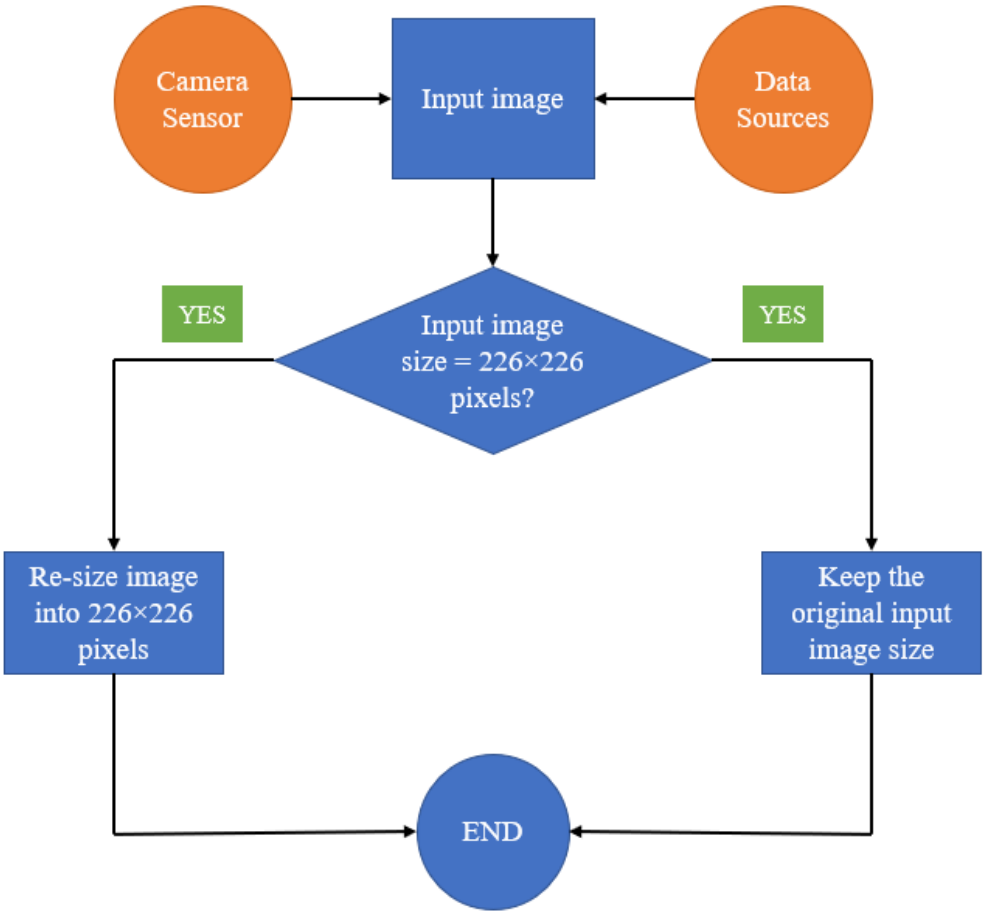
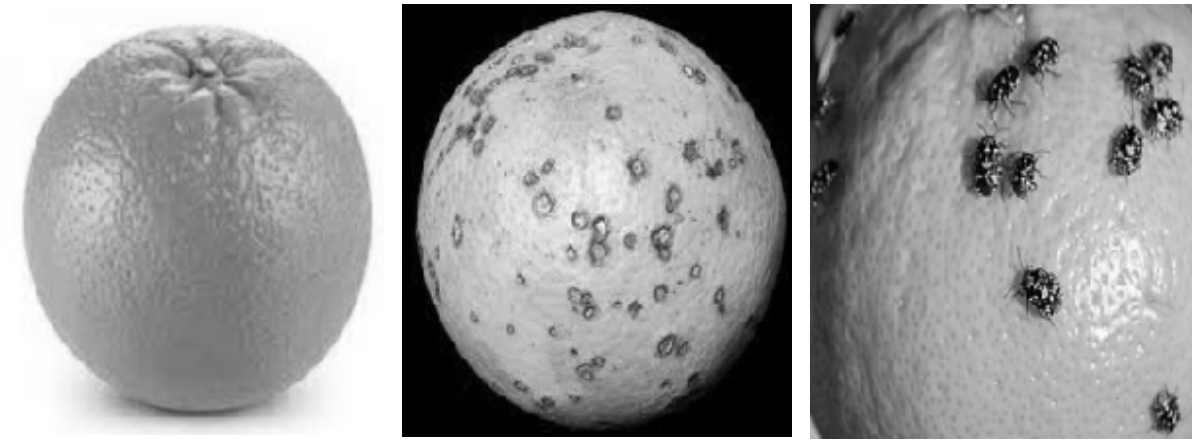


Figure 4-7: Block Diagram of the image re-sizing operation for the orange classifier

Fig. 4.7 shows the block diagram of a MATLAB code (shown in Appx. A1) that was used to import the input images into MATLAB's workspace and re-size them into the standard 226×226 pixels size as per the input size of GoogleNet which will be used as a classification model.

4.1.2.3. Image Grey-scaling

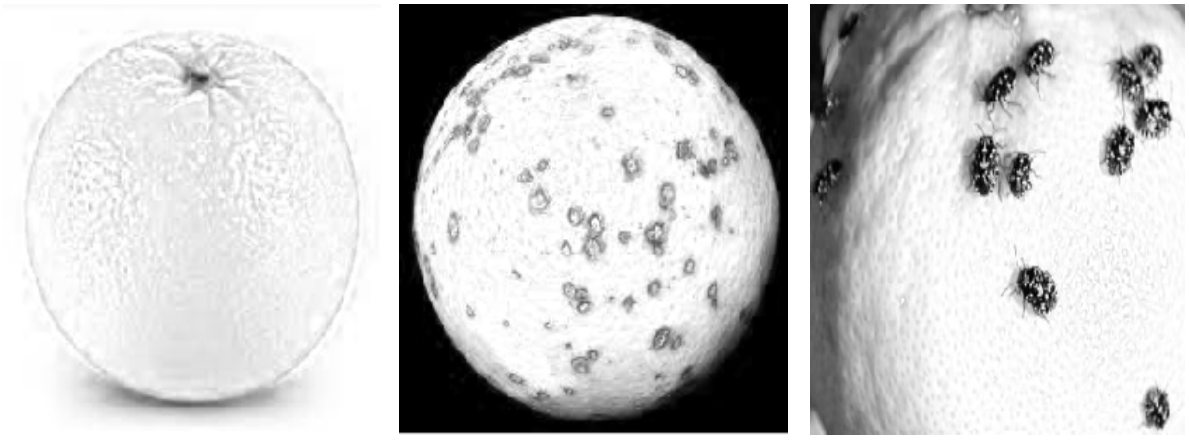
This stage eliminates all the colors in an image and replaces them with varying greyscale colors. This stage normally simplifies the algorithms required to further process images but is not always necessary, especially in cases where the color feature are important. The MATLAB code for generating greyscale images from the RGB images in Fig. 4.5 is shown in Appx. A2 and Fig. 4.7 shows the resulting greyscale images.



(a) (b) (c)

Figure 4-8: (a) A Greyscale healthy orange; (b) Black-rot affected orange greyscale image ; (c) Pest-infested orange greyscale image

As mentioned earlier, image grey-scaling is not always imperative in image pre-processing. This study has realized that in some cases, dissociating an RGB image into its 3 component color planes, performing thresholding on each of the RGB planes, and finally superimposing the results may give better image segmentation. In some cases, only 1 or 2 components might provide perfect contrast between the background and foreground, hence thresholding should be performed only on such components, and in this case, only the red and blue components, refer to Fig. 4.9. The MATLAB code for generating the individual RGB components from the RGB images in Fig. 4.5 is shown in Appx. A3 and Fig. 4.9 (a-c), (d-f), and (g-i) shows the corresponding red, green and blue components respectively.



(a) (b) (c)

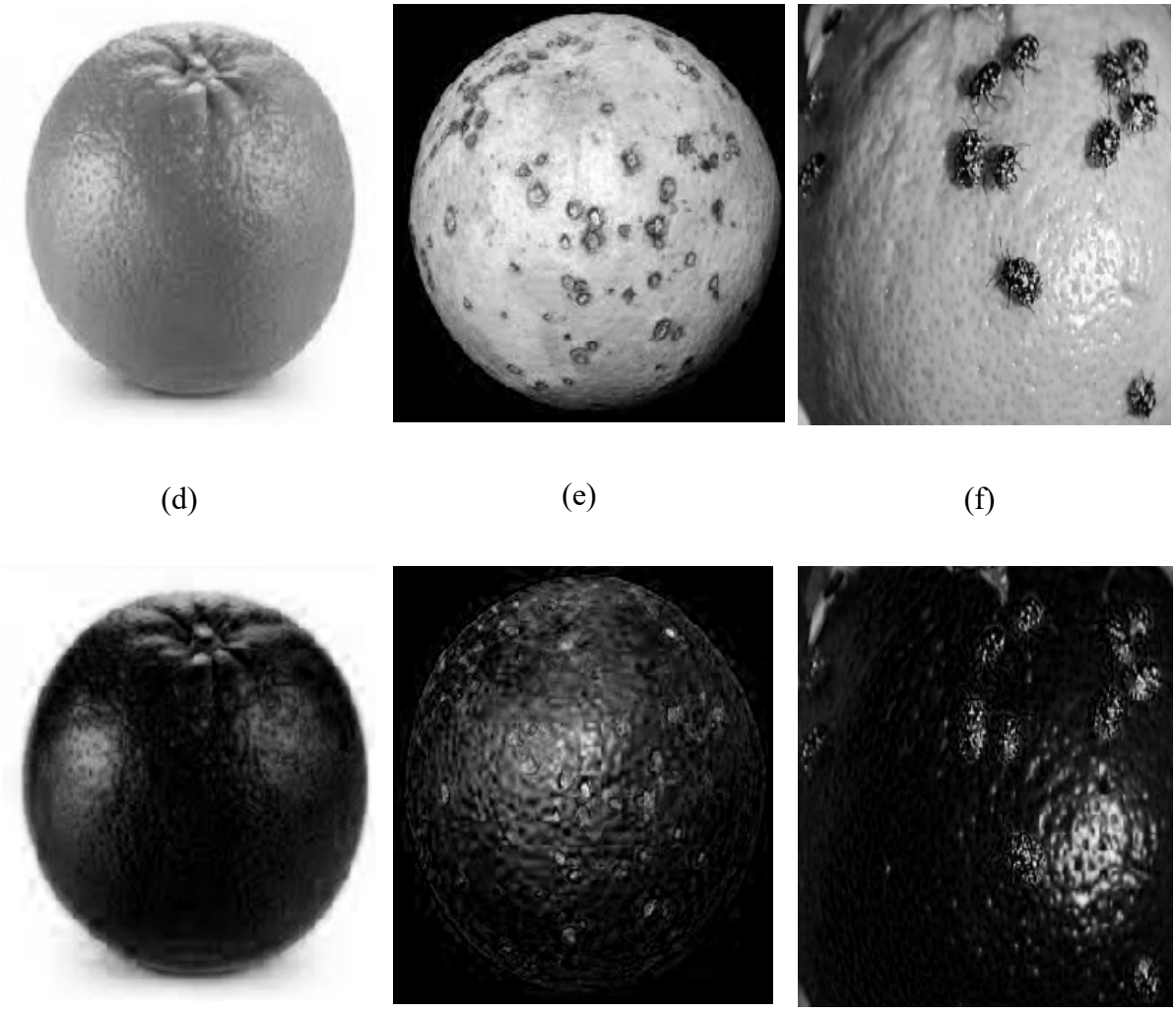
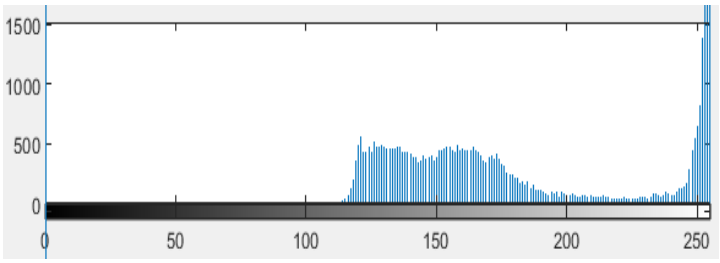


Figure 4-9: (a) A Greyscale healthy orange; (b) Black-rot affected orange greyscale image ; (c) Pest-infested orange greyscale image (d) A Greyscale healthy orange; (e) Black-rot affected orange greyscale image ; (f) Pest-infested orange greyscale image (g) Black-rot affected orange greyscale image ; (h) Pest-infested orange greyscale image; (i) Pest-infested orange greyscale image

4.1.2.4. Image Contrast enhancement

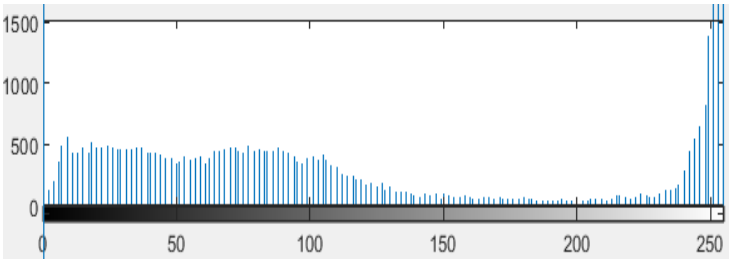
This stage is imperative to achieve a clear distinction between the foreground and background. This is achieved via two methods in this thesis, namely histogram equalization and pixel intensity stretching. Both these methods tend to spread the pixels of a greyscale image across all intensities ranging from 0-255 (sometimes mapped to 0-1) instead of being concentrated on smaller sub-range(s) [1]. The MATLAB code for contrast adjustment on the greyscale images in Fig. 4.7 is shown in Appx. A4, and Fig. 4.10 presents the results.

(a)


1500
1000
500
0
0
50
100
150
200
250

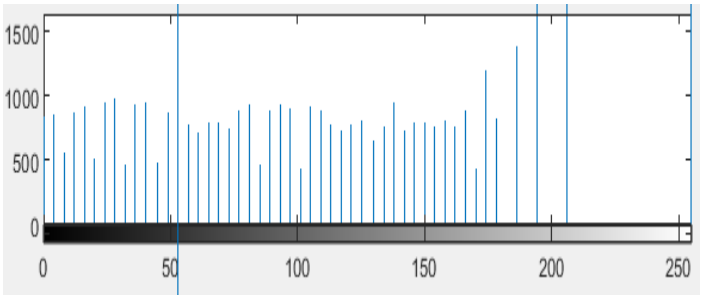
(b)

(c)

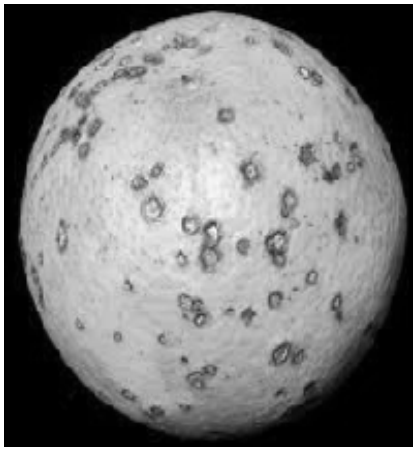

1500
1000
500
0
0
50
100
150
200
250

(d)

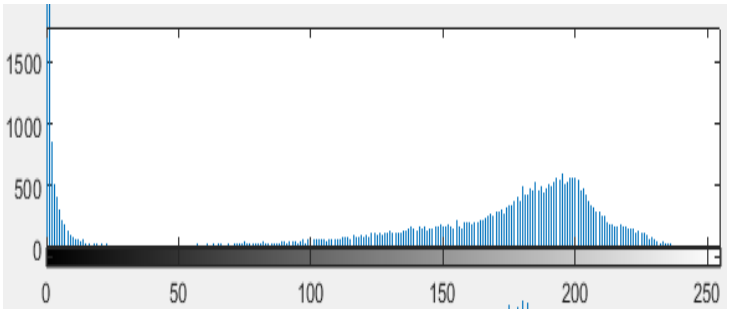
(e)


1500
1000
500
0
0
50
100
150
200
250

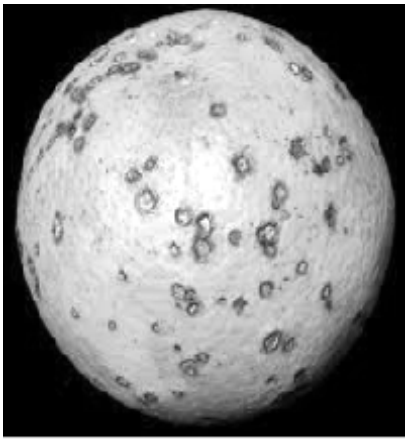
(f)



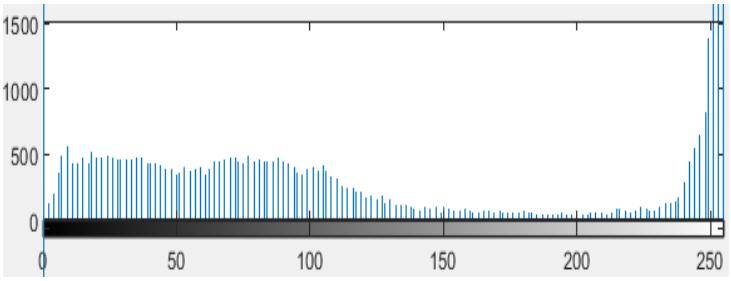
(g)


1500
1000
500
0
0
50
100
150
200
250

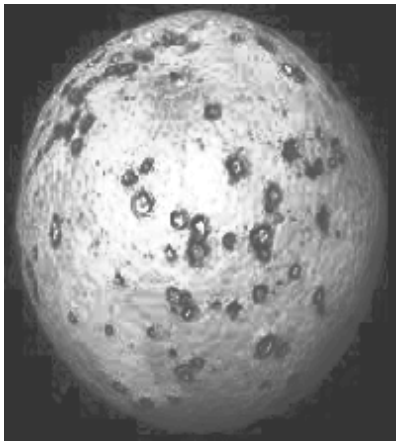
(h)



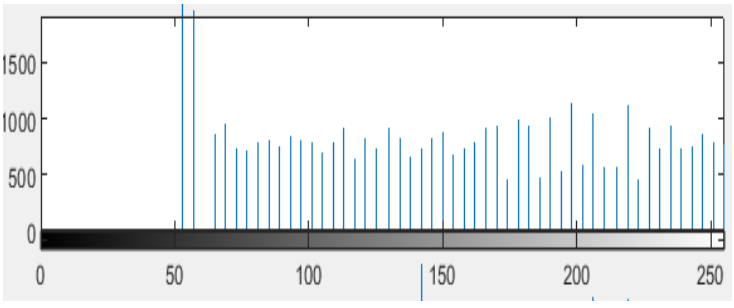
(i)


1500
1000
500
0
0
50
100
150
200
250

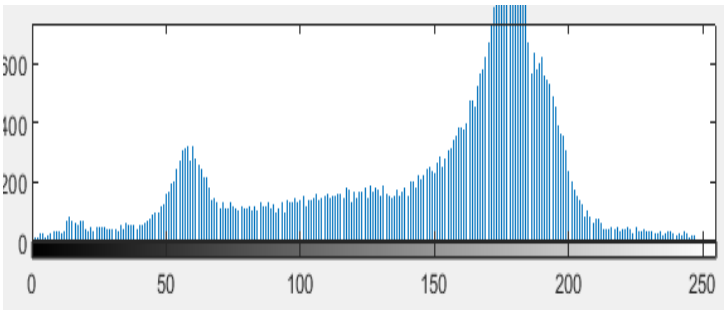
(j)



(k)

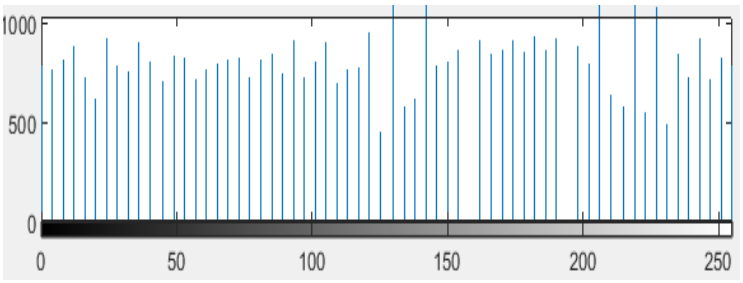

1500
1000
500
0
0
50
100
150
200
250

(l)



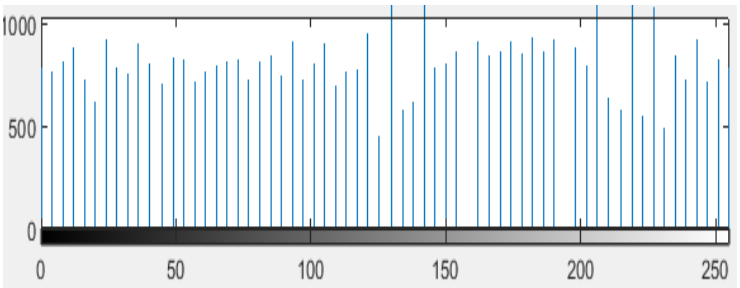
(n)

(m)



(p)

(o)



(r)

(q)

Figure 4-10.: (a) A Greyscale healthy orange; (b) Pixel histogram (for Fig. 4.9(a)); (c) An intensity-stretched greyscale healthy orange; (d) Pixel histogram (for Fig. 4.9(c)); (e) A histogram equalized greyscale healthy orange; (f) Pixel histogram (for Fig. 4.9(e)); (g) A greyscale black-rot infected orange;

(h) Pixel histogram (for Fig. 4.9(g)); (i) An intensity-stretched greyscale black-rot infected orange; (j) Pixel histogram (for Fig. 4.9(i)); (k) A histogram equalized greyscale black-rot infected orange; (l) Pixel histogram (for Fig. 4.9(k)); (m) A greyscale pest-infested orange; (n) Pixel histogram (for Fig. 4.9(m)); (o) An intensity-stretched pest-infested orange; (p) Pixel histogram (for Fig. 4.9(o)); (q) A histogram equalized greyscale black-rot infected orange; (r) Pixel histogram (for Fig. 4.9(q))

Fig. 4.10 (a, c & e) shows the greyscale images of a healthy orange for normal grey-scaling, pixel intensity stretching, and histogram equalization respectively; while Fig. 4.10 (b, d & f) shows the corresponding pixel distribution histogram. Fig. 4.10 (g, i & k) shows the greyscale images of a black-rot infected orange for normal grey-scaling, pixel intensity stretching, and histogram equalization respectively; while Fig. 4.10 (h, j & l) shows the corresponding pixel distribution histogram. Fig. 4.10 (m, o & q) shows the greyscale images of a pest-infested orange for normal grey-scaling, pixel intensity stretching, and histogram equalization respectively; while Fig. 4.10 (n, p & r) shows the corresponding pixel distribution histogram.

For normal image grey-scaling across healthy, black-rot-infected and pest-infested oranges, one can notice that the colour-intensity pixel distribution is clustered across a narrow range of colour-intensity values. This undermines a clear-cut contrast between a background and a foreground, hence necessitating the application of pixel intensity stretching and histogram equalization. Results presented in Fig. 4.10 provide a clear comparison of the two mentioned techniques and in this application, pixel intensity stretching offered an improved contrast and a better distribution of pixels across different colour-intensity values while histogram equalization excessively spreads the pixel intensity values beyond a point of clear contrast between a foreground and a background. The block diagram below summarizes the proposed optimized pre-processing procedure proposed in this research study.

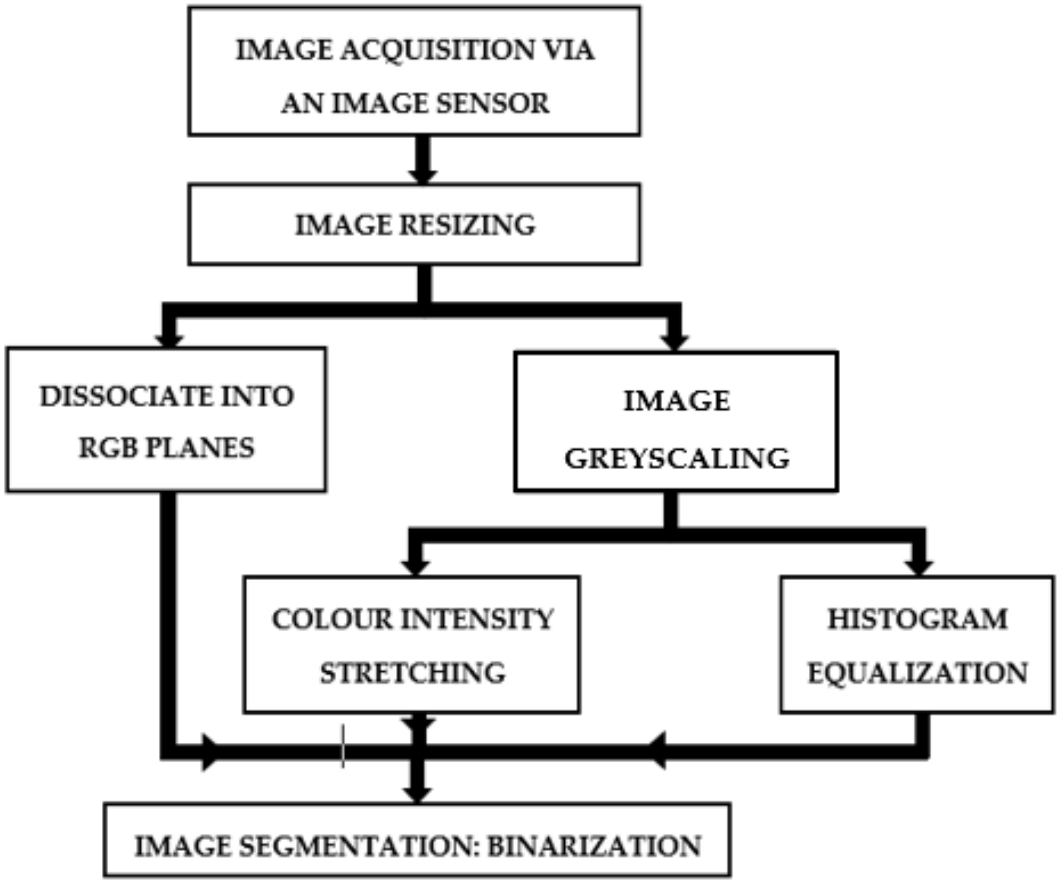


Figure 4-11: The proposed image pre-processing model for fruit disease detection systems (Proposal 1)

In the block diagram in Fig. 4.11, there are three possible paths through which image pre-processing can be achieved. The first goes as follows: Image acquisition via an image sensor, image re-sizing, dissociation of an image into RGB planes, and then image segmentation. The second goes as follows: Image acquisition via an image sensor, image re-sizing, image grey-scaling, pixel color intensity stretching, and image segmentation. The third and last goes as follows: Image acquisition via an image sensor, image re-sizing, image grey-scaling, pixel color intensity histogram equalization, and image segmentation. The path that yields better results is dependent on the properties of the images being processed and the particular application at hand.

4.1.2.5. Image segmentation

Five different thresholding techniques appear most frequently in literature, namely manual thresholding, Otsu, grab-cut segmentation, Hue-Saturation-Value (HSV), and watershed methods. However, this study will discuss the first three which operate on a greyscale image and omit the last two which operate on RGB images.

(i) Manual thresholding

Manual thresholding is the process of manually selecting a threshold value that generates the best segmentation between the foreground and background. This may involve several iterations while continuously adjusting between threshold values before landing on acceptable image segmentation. Pixel color intensity graphs and histograms are useful tools in implementing manual thresholding because they can give one a clue as to where to place their first attempt threshold value by analyzing the pixel distribution across different color intensities. This reduces the number of iterations before landing on a good segmentation and consequently reduces the computational time. Therefore, using this technique to segment the pixel color intensity stretched greyscale images in Fig. 4.9 (a, i, o) and pixel color intensity histograms in Fig 4.9 (b, j, p) are considered respectively.

- First iteration

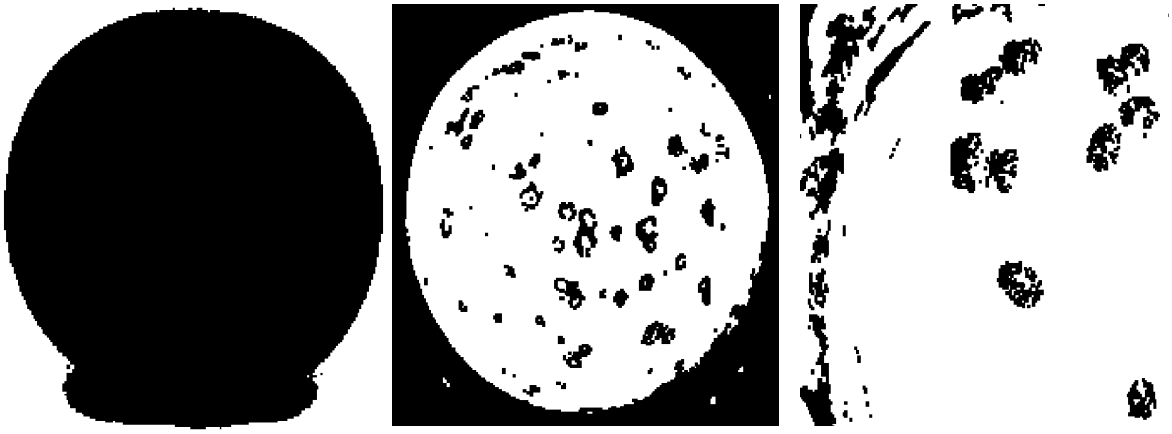
The objective is to select a threshold value that best segregates the features to be extracted from the rest of the image. To do so, one should consider the minimum colour-intensity of the features of interest and locate it on the greyscale found in the pixel colour-intensity histograms. Considering the pixel colour-intensity stretched greyscale images in Fig. 4.10 (a, i, o) and their corresponding pixel colour-intensity histograms in Fig. 4.10 (b, j, p), the thresholding values are 175, 160 and 85 respectively. These threshold values lie in a [0-255] and should be mapped to a [0-1] scale for MATLAB programming, and the corresponding threshold values are 0.69, 0.63 and 0.33 respectively. Fig. 4.12 shows the results of the selected threshold values.

(a) (b) (c)

Figure 4-12: (a) Healthy orange binary image (at a 0.69 threshold); (b) Black-rot affected orange greyscale image (at a 0.63 threshold); (c) Pest-infested orange greyscale image (at a 0.33 threshold)

- Last iteration

The first iteration resulted in binary images, with noise or unnecessary small objects being segmented into the foreground instead of the background, as seen in Fig. 4.12 (a-c). Therefore, the threshold values were incremented in +0.1 steps until the discrimination of the unwanted feature, until a good segmentation was achieved. The final selected threshold values are 0.78, 0.72 and 0.62 for healthy, Black-Rot affected and pest-infested orange greyscale images respectively, and the results are shown in Fig. 4.13 (a-c).



(a) (b) (c)

Figure 4-13: (a) Healthy orange binary image (at a 0.78 threshold); (b) Black-rot affected orange greyscale image (at a 0.72 threshold); (c) Pest-infested orange greyscale image (at a 0.62 threshold)

(ii) Otsu's segmentation

By definition, Otsu's segmentation method is an image binarization algorithm that automatically computes an optimum color-intensity threshold value that best segregates the foreground from the background [12]. This is achieved by considering all different color intensities present in an image, each as a possible threshold value, and computing a color intensity variance for the pixels both in the fore- and background. The optimum threshold is the one presenting the minimum pixel color-intensity variance in the fore- and background. Assume an image with a foreground f and background b [13]. Then, the variance between the foreground and background σ is mathematically defined by Eqn. (4.1), where W_f and W_b are weighted sums of the foreground and background pixels respectively; while σ_f and σ_b are the variances between the foreground and background pixels respectively [12, 13]. Otsu's algorithm then iterates through all possible color-intensity thresholding values while computing Eqn. (4.1) for each iteration [12]. Any thresholding value yielding the minimum variance between the foreground

and background will be the final selected thresholding value. MATLAB has a built-in Otsu segmentation algorithm that can be implemented using the piece of code shown in Appx. A5, and the resulting binary images are shown in Fig. 4.14.

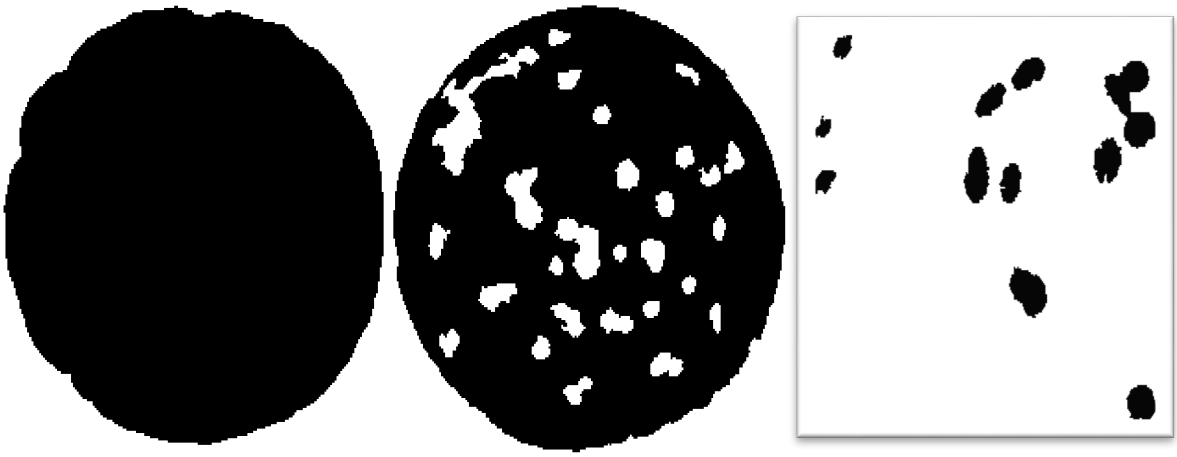
1423 $$\sigma = W_f \sigma_f + W_b \sigma_b$$ 4-1

(a) (b) (c)

Figure 4-14: Healthy orange binary image (by Otsu's method); (b) Black-rot affected orange greyscale image (by Otsu's method); (c) Pest-infested orange greyscale image (by Otsu's method)

(iii) Grab-cut segmentation

Grab-cut is a type of image segmentation method that operates through graph cuts. This type of segmentation algorithm is usually applied in computer vision equipment such as x-ray scans, computerized tomography (CT) scans, magnetic resonance imaging (MRI) scans, face recognition security systems, etc. [1-3]. In this segmentation algorithm, an operator marks the target object on the sample image that they wish to segment as a foreground. They also mark the background, and the algorithm estimated the color-intensity distribution of both the back and foreground using the Gaussian Mixture Model (GMM). GMM is a statistical function that estimates the probability of either falling in the back or foreground cluster of each pixel in an image. Fig. 4.15 shows the binary images of figure healthy, black-rot-infected and pest-infested oranges formed through the grab-cut segmentation algorithm.



(a) (b) (c)

Figure 4-15: (a) Healthy orange binary image (at grab-cut segmentation); (b) Black-rot affected orange greyscale image (at grab-cut segmentation); (c) Pest-infested orange greyscale image (at grab-cut segmentation)

Manual and gab-cut image segmentation yields the best results, while Otsu's segmentation methods result in poor segmentation in this example because the resulting binary image has high noise content. It is also important to note that the quality of a sample image is imperative for subsequent good image segmentation. Hence, if the sample images are captured manually, aspects such as lighting conditions, background colour, and neighbouring objects should be taken into account. Good image segmentation informs a good image feature extraction.

4.1.2.6. Image feature extraction

Feature extraction in the context of machine learning-based image processing refers to a process of marking the relevant features (foreground) while discriminating the rest (background) in an image that will consequently assist a machine learning model to classify an image [13]. Features in images can be categorized into 3 main classes, namely colour, texture and size features as detailed in Chapter 2. Several feature extraction models have been proposed in the literature. These include the likes of colour moments (CM), colour coherence vector (CCV), colour correlograms (CC), dominant colour descriptor (DCD), colour structure descriptor (CSD), scalable colour descriptor (SCD), ray-level co-occurrence matrix (GLCM), grey level run length matrix (GLRLM), local binary patterns (LBP) and gradient-based features (GBF) [13, 14, 15].

In machine learning, feature extraction and classification occur separately. That is, a model trainer has to pre-select the features to be extracted and feed them to a machine learning classification algorithm as inputs, and that is where all the above-mentioned feature extraction models are applied. However, in deep learning, feature extraction and image classification occur automatically within a deep learning neural network. That is, a neural network learns on its own, during training, which features to extract to produce the output at a certain accuracy level. Refer to Fig. 4.16 for different feature extraction techniques in general machine learning versus deep learning algorithms.

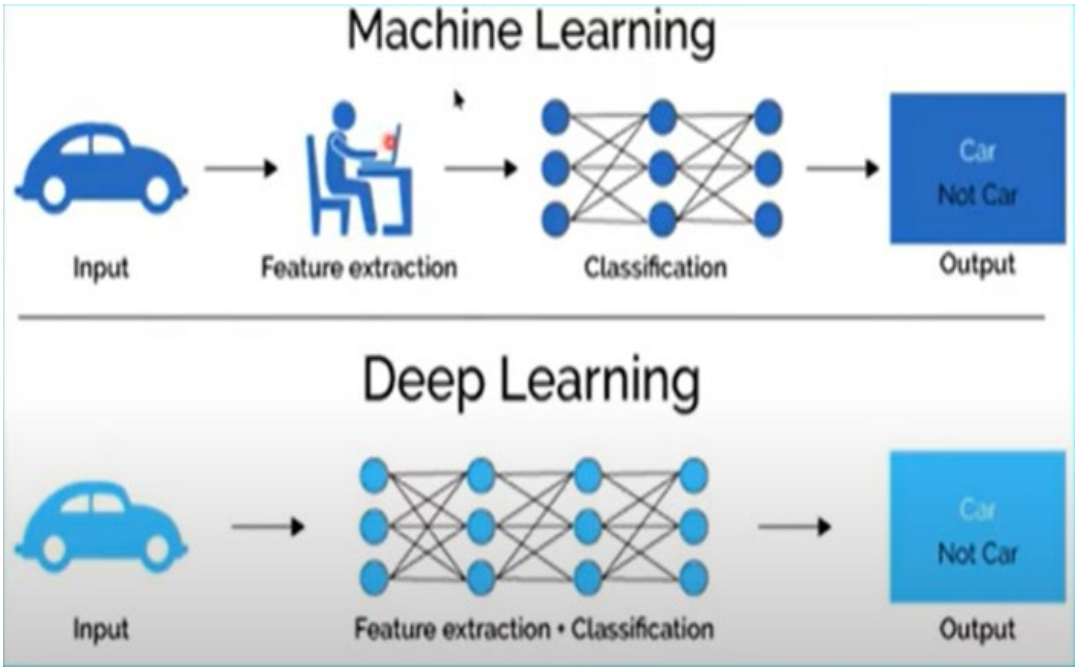


Figure 4-16: Feature extraction in general machine learning vs. deep learning models

This study utilizes a technique called transfer learning to classify healthy vs. diseased fruits. Transfer learning is a process of re-training an existing deep learning model to solve a different problem. This process involved modifying selected layers in a neural network to suit the new set of data at hand. Therefore, for this particular study, feature extraction will be performed automatically and within the neural network of choice.

4.1.3. The healthy-black rot-affected orange classification model

This study utilizes an open library convolutional network called GoogleNet to perform classification operations. Fig. 4.17 shows a generalized architecture of a convolutional neural

network and its major parts. A CNN shown below is separated into 4 main sections; the input, feature extraction, image classification, and output section.

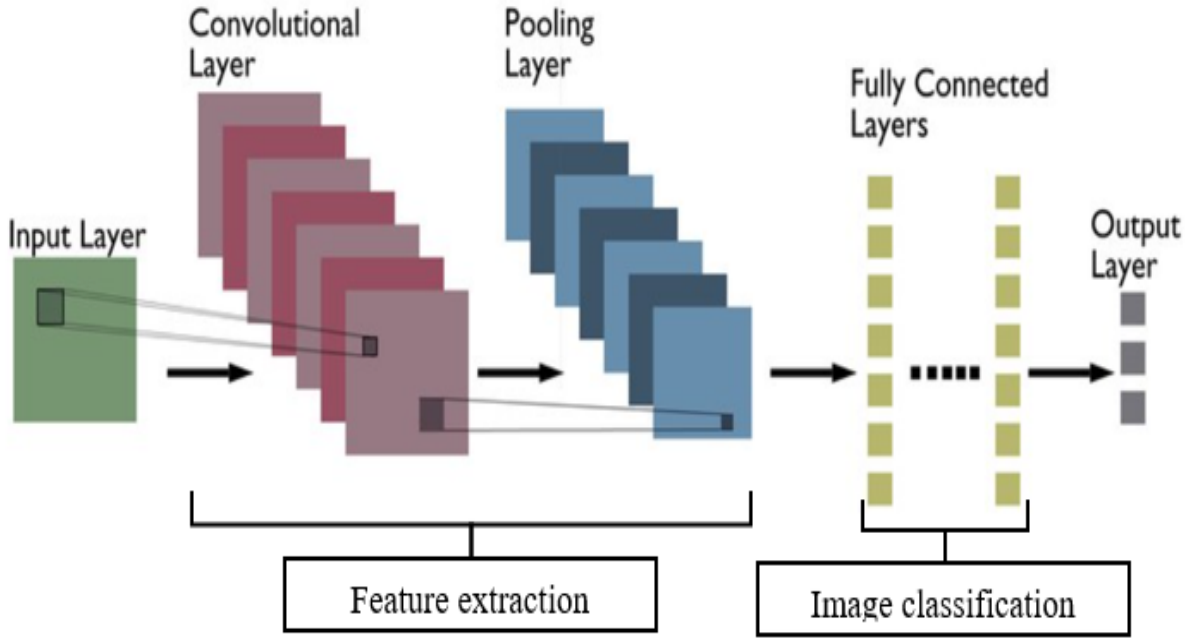


Figure 4-17: General architecture of a CNN

The input image is transferred from the input layer into a feature extraction or feature learning section. The learned features are then transferred to the fully connected layers for classification and lastly, the predicted output is transferred to the output layer. At the input layer, the number of input nodes is equal to the total number of pixels in the input images. The feature learning layer consists of a pile of convolution (plus the activation function) and pooling layer pairs. These layer pairs can be compared to digital filters, the convolution layers converge the images utilizing the convolution technique while the pooling layers combine the spatial neighboring pixels into singular pixels, hence reducing the dimensions of a sample image. The classification section which consists of fully connected layers is then trained (refer to Chapter 3) to classify different images according to different classes.

4.1.3.1. Training GoogleNet to classify between healthy and black-rot affected oranges

This section designs a deep learning classifier that classifies healthy and black-rot affected oranges. This is achieved through the utilization of the MATLAB deep learning toolbox. Here, only the training and validation of the classifier will be covered, while the next section will be

covered in the coming section. The first step towards training a neural network is to create a dataset. The dataset should be dissected into training and validating data.

- Creating a dataset

The training and validating datasets were sources from an open library website called Kaggle, the citrus and citrus diseases datasets respectively. The dataset folder has two sub-folders in it called 'healthy oranges' and 'black-rot affected oranges'. Each folder contains 160 corresponding images. This dataset is loaded into MATLAB's current folder before proceeding to the next step, as shown in Fig. 4.18. During the operation of this neural network when training and validation are complete, the model will be getting input images through snapshots from a live video feed from a Macally webcam in Fig. 4.4.

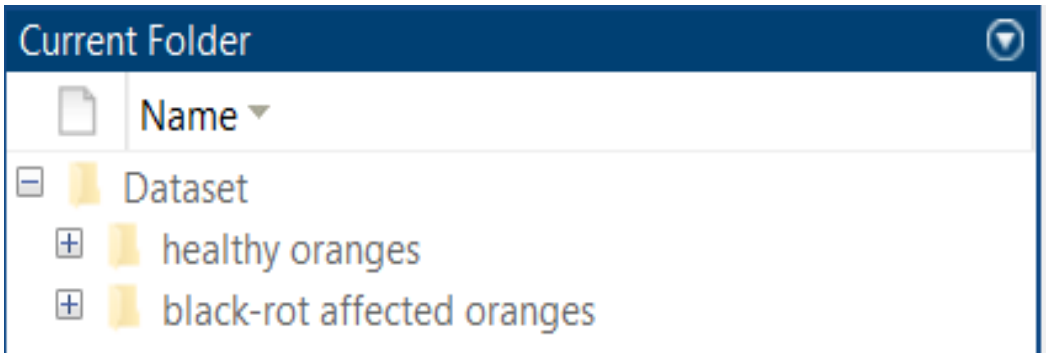


Figure 4-18: Dataset loaded into MATLAB's current folder

- Training GoogleNet

The second step is training a GoogleNet neural network. It is important to note that if other machine learning models were utilized for this classification problem other than deep neural networks, the image pre-processing, segmentation and feature extraction would have had to be performed manually before training a classification model. However, since this project is utilizing deep neural network transfer learning and these tasks are performed inside a neural network, only minor image pre-processing tasks are imperative in this case, such as image dimension re-sizing to match the size of an input layer of GoogleNet. The MATLAB code used for training a GoogleNet network is shown in Appx. A6.

This MATLAB program first exports the dataset with its sub-folders from MATLAB's current folder to its workspace while setting folder names to classification labels. A pre-trained

GoogleNet neural network is also loaded into the workspace. The image dataset is then re-sized into 224×224 pixels (the input size of GoogleNet as seen in layer 1 in Fig. 4.19) and dissected into 70% and 30% training and validation data respectively. As seen in Fig 4.19, the original feature learner (layer 142) and classifier (layer 144) of Googlenet are originally designed to classify 1000 different classes of images. The feature learner and classifier are re-named to orange feature learner and orange classifier as shown in Fig. 4.20, and GoogleNet is trained according to a specified training rule and options.

ANALYSIS RESULT

	Name	Type	Activations
1	data 224×224×3 images with 'zerocenter' nor...	Image Input	224(S) × 224(S) × 3(C) × 1(B)
142	loss3-classifier 1000 fully connected layer	Fully Connected	1(S) × 1(S) × 1000(C) × 1(B)
144	output crossentropyex with 'tench' and 999 oth...	Classification Output	1(S) × 1(S) × 1000(C) × 1(B)

Figure.4-19: The input, feature learner and classification layer of GoogleNet before transfer learning

ANALYSIS RESULT

	Name	Type	Activations
142	Orange Feature Learner 2 fully connected layer	Fully Connected	1(S) × 1(S) × 2(C) × 1(B)
144	Orange Classifier crossentropyex	Classification Output	1(S) × 1(S) × 2(C) × 1(B)

Figure.4-20: The feature learner and classification layer of GoogleNet after transfer learning

During training, the training progress was observed during the specified 10 different epochs (1 iteration per each epoch), as shown in Fig. 4.21. As training progresses, the training efficiency increases to maximum by the 4^{th} epoch, and the validation efficiency increases steadily as epochs increase, which means that the model becomes more accurate as it processes more and more images. Similarly, the training and validation losses decrease as epochs increase which tells that the model is learning correctly. The validation efficiency and learning duration were 98.58% and 163 seconds respectively (refer to Fig. 4.22) at the end of the last epoch.

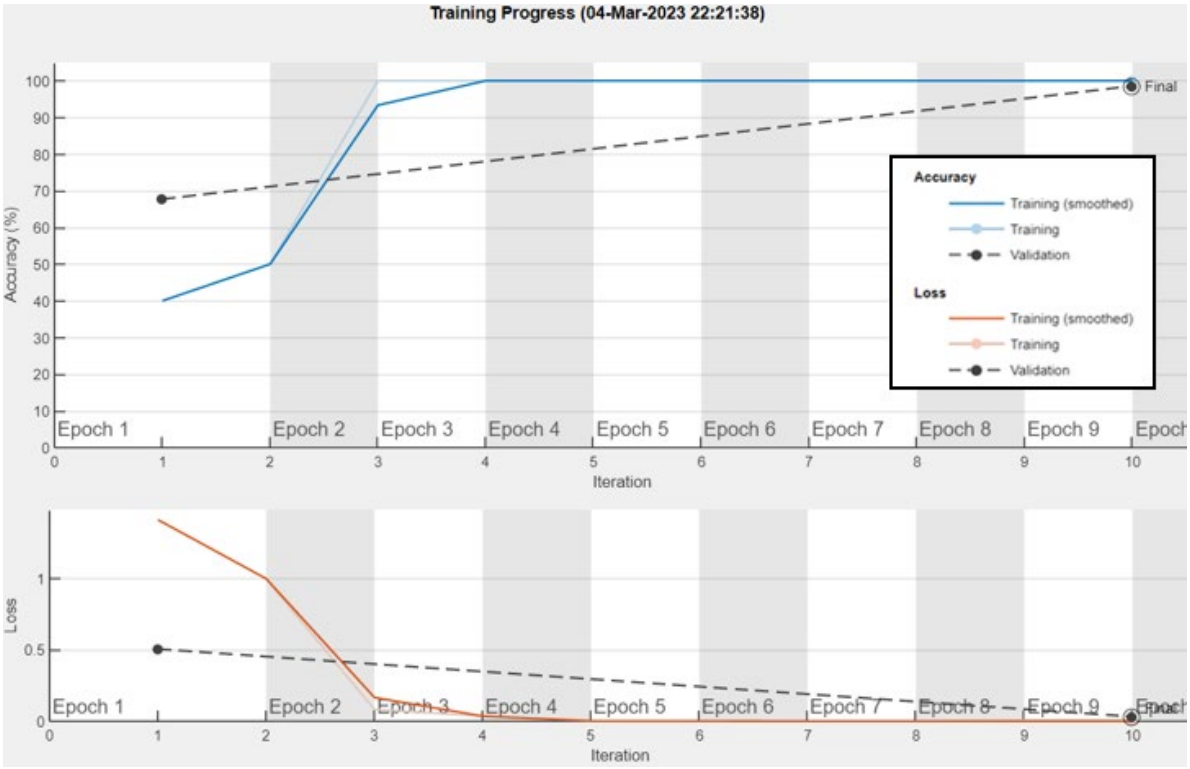


Figure 4-21: Orange classifier's training progress against an increasing number of epochs

Results

Validation accuracy:	98.58%
Training finished:	Max epochs completed

Training Time

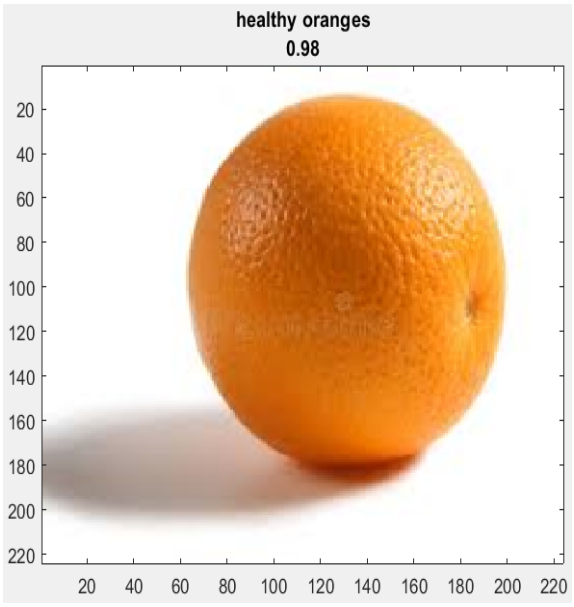
Start time:	04-Mar-2023 22:21:38
Elapsed time:	2 min 3 sec

Figure 4-22: Orange classifier's learning duration and efficiency

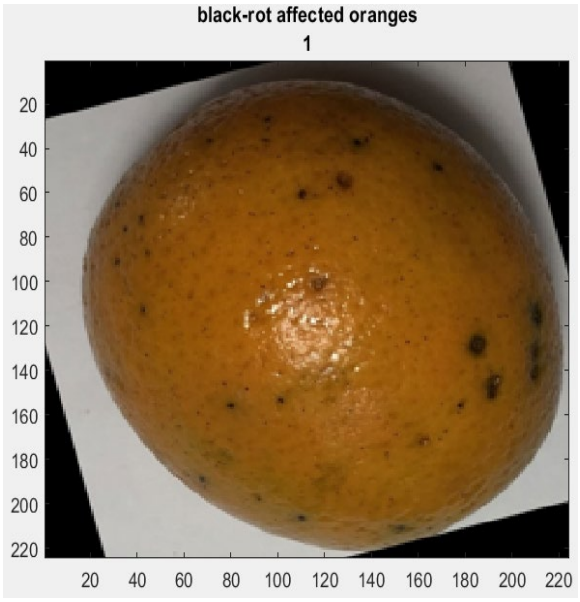
4.1.3.2. Testing an orange classifier

At this point, the design and validation of a neural network orange fruit classifier are complete. Now it is time to test this model on a practical example to see if it can classify correctly between healthy and black rot-affected oranges. This will be achieved by acquiring a sample image and passing it to an already trained model in the previous section for classification. The MATLAB code that is shown in Appx. A7 was used to achieve the purpose of this section.

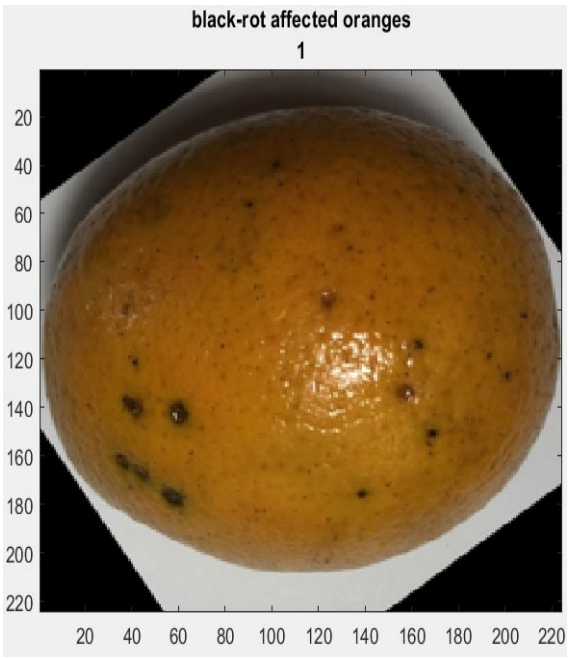
This code takes snapshots from a Macally webcam live feed video, re-sizes the images into 224×224 pixels, and classifies these images according to their classes. The classified images are then printed in image windows with their predicted class and probability of that prediction. However, since there are no sample arranged fruits at the time of documentation of this thesis, 6 test images (3 healthy and 3 blackout-affected oranges) were downloaded from Kaggle and passed through the orange classification model for classification, and the results are presented in Fig. 4.23 below.


healthy oranges
0.98

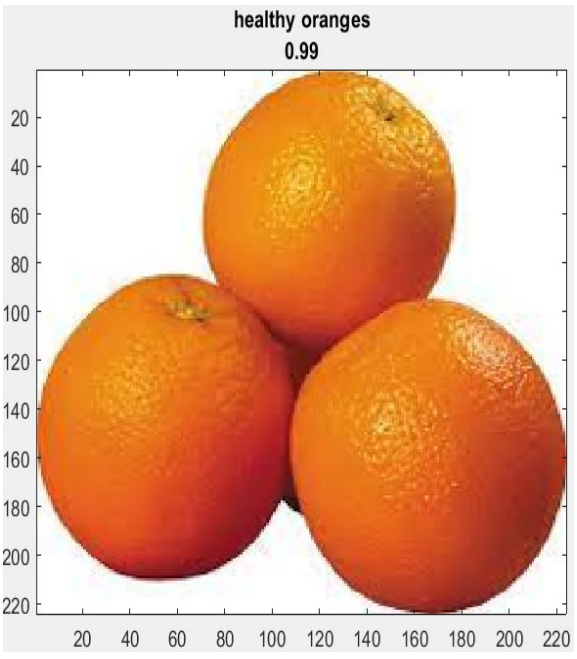
(a)


black-rot affected oranges
1

(b)

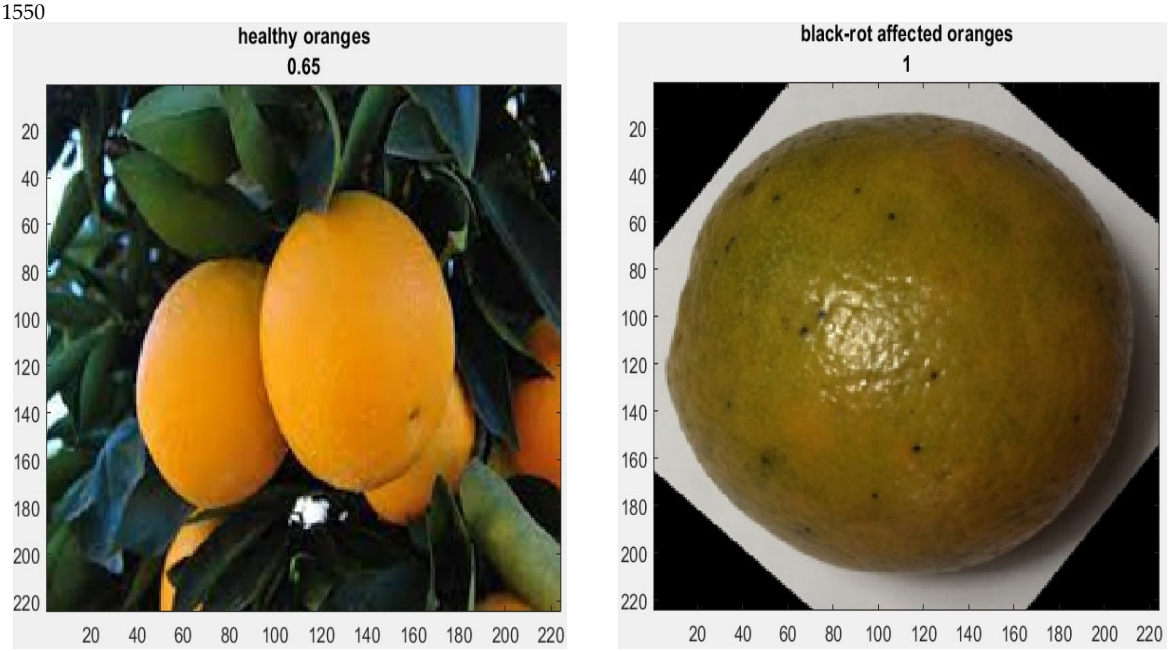

black-rot affected oranges
1

(c)


healthy oranges
0.99

(d)

1550



(e) (f)

Figure 4-23.: (a) 1st Healthy orange classified correctly @ 98% probability; (b) 1st Black-rot affected orange classified correctly @ 100% probability; (c) 2nd Black-rot affected orange classified correctly @ 100% probability; (d) 2nd Healthy orange classified correctly @ 99% probability; (e) 3rd Healthy orange classified correctly @ 65% probability; (f) 3rd Black-rot affected orange classified correctly @ 100% probability

4.3. Conclusion

In this chapter, all the principles of a fruit disease classifier, more particularly an orange classification model, were conceptualized and implemented from image acquisition to training and testing a classification model. An improved image pre-processing technique was proposed to achieve better image segmentation and hence good feature extraction. Manual image pre-processing techniques and feature extraction is required for all other machine learning models except for deep neural networks where feature learning occurs automatically within the network itself. A GoogleNet neural network was successfully trained to classify between healthy and black rot-affected oranges as per the set of results presented in Fig. 4.23. The orange classifier was able to classify 6 orange images correctly with a probability ranging from 98-100%, which is an excellent result. The only outlier was an image of healthy oranges (Fig. 4.23 (e)) which had a large portion of green leaves in the background, hence destructing a classifier. This problem could have been avoided by clipping and cropping the image or capturing images such that minimum objects can be seen in the background. Alternatively, this problem can be solved

by expanding the training dataset to include more similar images where leaves are present in the background. In that way, a classifier model will learn how to classify such images with high confidence intervals.

The coming Chapter (Chapter 5) develops several improvements to the classical plant disease classification model, aiming to improve their classification accuracy and efficiency. For each improvement proposal made, the conceptualization, modelling, testing aspects and comparative evaluation relative the traditional orange classifier (developed in this Chapter 4) are discussed.

CHAPTER 5: THE PLANT DISEASE CLASSIFICATION MODEL IMPROVEMENT PROPOSALS

5.1. Introduction

This chapter proposes three improvements to the traditional plant disease classification models, aiming to improve the classification accuracy, efficiency and general credibility of these systems. The three different plant disease classification model topologies proposed in this chapter are:

- A 3-Dimensional (3-D) Image Input-Based Plant Disease Classification Model;
- A Multi-Camera Image Input-Based Plant Disease Classification Model; and
- A Hybrid Fruit Disease-Quality Monitoring and Sorting Model (H-DQMS Model)

In the upcoming sections, each of the above models is conceptualized, modelled, tested (via MATLAB's deep learning toolbox) and evaluated against the traditional orange classifier presented in Chapter 4.

5.2. A 3-Dimensional Image Input-Based Plant Disease Classification Model (Proposal 2)

The study has realized that almost all the plant disease classification models proposed in the literature have single input cameras and take a single snapshot image of a sample plant part (fruit, root, stem, leaf, flower, etc.) to perform a classification. This implies that such classification models have a 2-Dimensional (2-D) 'view' of a sample plant part. The purpose of a 3-Dimensional image input-based plant disease classification model is to grant the classification model a 3-D 'view' of a sample plant part for more efficient, accurate and reliable classification. The sub-sections that follow discuss the problem statement which gave birth to the proposed system, its conceptualization, modelling, and testing.

5.2.1. Problem Statement

The classification accuracy and efficiency of disease detection models can be compromised by several factors. These factors can range from the size to the variability of the training dataset. If a classification model is trained with fewer data samples, the model will learn fewer features during the training process. Similarly, if a classification model is trained utilizing sufficient but

similar data samples, the model will also learn a few features during the training process. When the training is complete and the model is presented with an object (fruit in the case of this study) that has slightly different features relative to the training dataset, the classification accuracy and efficiency are greatly impacted.

Defects on the fruits are not always distributed evenly across the whole surface area of fruits. This implies that the signs and symptoms of fruit diseases and nutrient deficiencies might appear on certain parts of fruits while the rest appears healthy. During the operation of a classification model, if the sample fruit is orientated such that the defective parts of a fruit are not in the line of sight or the view of a camera sensor, then that fruit can be wrongly classified as a healthy fruit, hence compromising the accuracy of a classification model. Very few publications that the author has reviewed so far (including the citations of this study) utilize multiple infeed cameras, and most related work publications utilize a single infeed which might not be able to capture the vital details of sample fruits due to their general spherical nature.

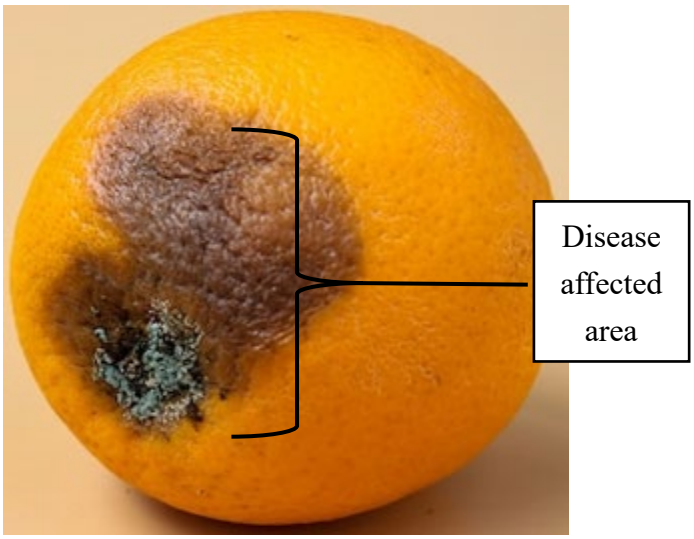


Figure 5-1: A sample fruit with uneven distribution of the disease-infected surface area [6]

Fig. 5.2 shows a sample orange fruit that has the symptoms of a black rot disease on the small section of its surface area (less than 30% of the total surface area), while the rest of the fruit appears healthy. If this fruit was orientated such that this rotten part of the fruit is 180˚ opposing the camera sensor, the model would have classified this orange as healthy. This may be a challenge in high throughput automated production lines where a model classification is processing multiple samples at a unit of time because samples can come at any orientation into the classification model.

5.2.2. Conceptualizing the proposed system

The distinction between the model proposed in this study from the previously existing models is that the proposed model seeks to capture the right-round surface (in the form of multiple images captured at different angles) of a sample fruit being classified. The proposed model then performs a classification on each image and consolidates the results. The final classification is then made using a specific protocol still to be discussed.

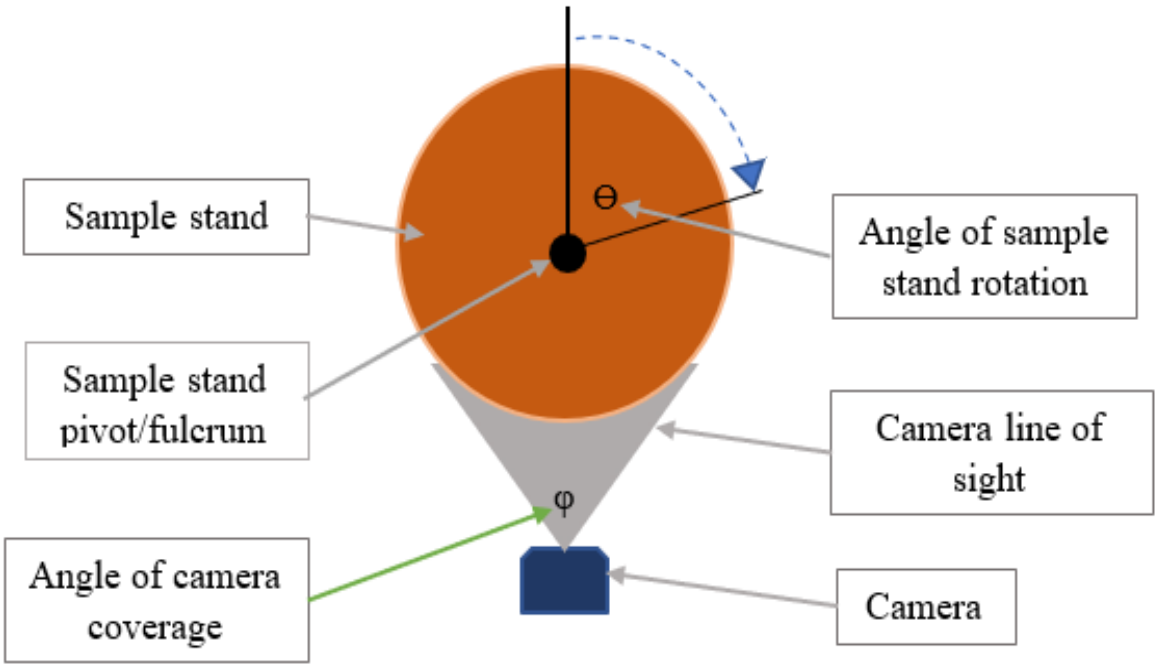


Figure 5-2: Top view of the proposed model image input system

Fig. 5.2 shows an image input system of the proposed system comprising a camera/image sensor and a sample fruit stand where a fruit under evaluation is placed. The camera's line of sight, which is the maximum surface that a camera sensor can 'see' and capture in an image, is directly dependent on the distance (d) between the camera and the fulcrum of the image stand. If d increases, the amount of circumference (C) a camera can cover when assuming fixed φ increases (as shown by Eqn. (5.1)), and the opposite is true. However, d has a limit that is determined by the characteristics of a camera/image sensor. Maximum range (MR) is the maximum distance between a camera and the object, which allows for the capturing of a 'good'

image. The amount of circumference covered by the image sensor's line of sight approaches half of the full circle's circumference ($2\pi r/2$) if d approaches MR (r is the radius of a sample fruit stand), refer to Eqn. (5.2).

$$d \propto C \tag{5-1}$$

$$\lim_{d \to MR} C(d) \leq \pi r \tag{5-2}$$

During the operation of the proposed model, the sample is placed on the sample stand and the first input image is captured. The sample stand then turns the sample Θ degrees clockwise and the second input image is captured. The same circle is repeated n times until the full circumference of a sample fruit is covered. The proposed model performs classification on each sample image captured, sums all the results, and makes a final classification judgment. Refer to Section 3.5 where 'Classify (CLF)' is a MATLAB deep learning function that has been trained to classify a certain fruit disease (refer to Section 5.3.).

$$n \geq \frac{2\pi}{\varphi} \tag{5-3}$$

$$input\_images = image_1 + image_2 + \ldots + image_n + image_{n+1} \tag{5-4}$$

$$f(input\_images) = \sum_{1}^{n} CLF(image_i) \tag{5-5}$$

The number of turns a sample stand should make while the image is captured in every turn can be obtained by dividing the total sample circumference by the camera's angle of coverage (refer to Eqn. (5.3)). There are then $n+1$ input images evaluated before the final classification f (input images) can be achieved as shown in Eqn. (5.4). And finally, each input image is classified independently while the classification results are summed together as shown in Eqn. (5.5). The final classification is made according to the following protocol:

- If at least 1 input image is classified as infected by a disease, set the final classification to a 'diseased sample'.
- If all the input images are classified as healthy, set the final classification to a 'healthy sample'.

Let the Classification Accuracy (*CA*) and the Classification Mean Time (*CMT*) be the correctness with which the machine classifies a disease and the total time it takes to classify a single sample respectively. Then, the *CA* is directly proportional to n but inversely proportional to the *CMT* as per Eqn. (5.6). In theory, as n approaches infinity, *CA* approaches 100% (as per Eqn. (5.7)), but *CMT* also approaches infinity, which is undesirable. Therefore, all these parameters should be optimized in consideration of the camera sensor characteristics.

$$CA \propto \frac{n}{CMT} \tag{5-6}$$

$$\lim_{n\to\infty} CA\,(n) \rightarrow\ 100\% \tag{5-7}$$

5.2.3. Modelling of the Proposed System

This sub-section develops a deep learning 3-D input plant disease detection model that distinguishes between healthy oranges and those that have black rot. This is accomplished by using the MATLAB deep learning toolbox. The approach suggested in this proposal will be illustrated using orange fruits and the black rot disease as in Chapter 4. Therefore, the creation of a training dataset, training and validation of the proposed model will be similar to that of a classical orange classifier presented in Chapter 4 and hence will be omitted. However, the operation of the proposed system differs, and is therefore discussed in the next section.

5.2.4. Testing the Proposed Model

This sub-section discusses the testing of a 3-dimensional image input-based plant disease classification model. It is now time to put this model to the test in a real-world scenario to see if it can distinguish between healthy oranges and those that have black rot. Unfortunately, the

authors did not have access to a stand-alone microprocessor at the time they presented their work. As a result, the training and testing MATLAB software was run on a Lenovo ThinkPad, which served as the host microprocessor. The authors also had not designed an automatic sample stand, discussed in Section 5.2.2. Hence, for demonstration purposes, a sample was turned manually to vary Θ. The camera was positioned at a distance d from the same such that φ is 180 degrees. This implies n equals 1 to cover the whole circumference of a sample orange and the number of input pictures equals $2\ (n+1)$. The MATLAB code used to acquire the input images and test the proposed model is shown in Appx. A8.

This MATLAB code takes two snapshot input images from a single sample orange from 180 degrees opposite side, classifies them individually, and consolidates the results to make the final classification according to the protocol indicated in Section 5.2.2. Three different orange samples were collected to test the model operation, a healthy orange and two diseased oranges where black rot symptoms have spread evenly throughout the whole orange surface area and on only a small surface area on the other. The classification results of these samples are presented in Fig. 5.3-5.6.

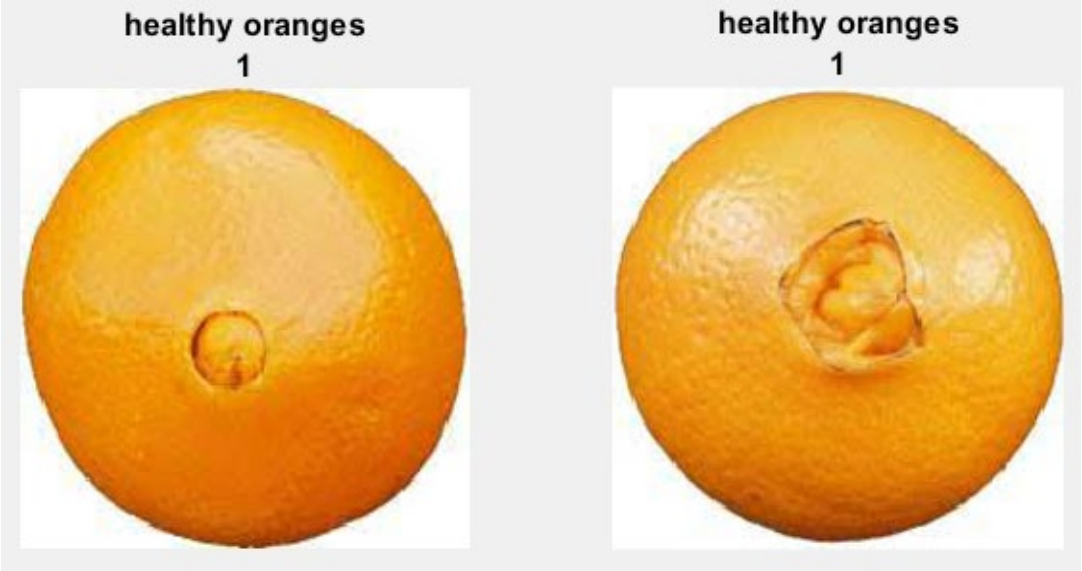


Figure 5-3: Healthy orange images from two opposite angles

```
Command Window
>> RealTimeFunction
Healthy Orange Detected!
Final Classification = Healthy orange!
```

Figure 5-4: Final classification results for a healthy orange

Figure 5-5: Diseased orange images from two opposite angles for an uneven black-rot distribution

```
Command Window
>> RealTimeFunction
Diseased Orange Detected!
Final Classification = Black rot affected orange!
```

Figure 5-6: Final classification results for a diseased orange with an uneven black-rot distribution

Figure 5-7: Diseased orange images from two opposite angles for an even black-rot distribution

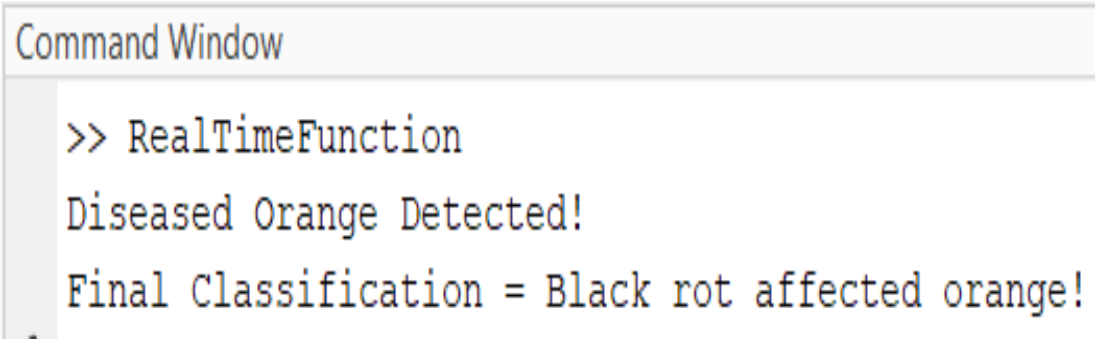


Figure 5-8: Final classification results for a diseased orange with an even black-rot distribution

Fig. 5.3 displays two images of a healthy orange taken from two different opposite angles. Each image was accurately identified as a healthy orange. The final categorization results for this sample, which are obtained by combining the findings from each angle, are shown in Fig. 5.4. Similarly, Fig. 5.5 displays two photos of a diseased orange (with unequal distribution of black rot symptoms) that were taken from two different opposite viewpoints, and one of them was classified as healthy while the other as a diseased orange. The final categorization outcome for this sample is the 'black rot affected orange' as per the protocol in Section 5.2.2, as shown in Fig. 5.6. Lastly, Fig. 5.7 displays two photos taken from two different viewpoints of a sick orange (with an even distribution of black rot symptoms). The final classification is displayed in Fig. 5.8, also as 'black rot affected orange'.

5.2.5. Evaluation and Comparison of a 3-Dimensional Image Input-Based Plant Disease Classification Model Relative to a Healthy-Black Rot Affected Oranges Classification Model

In this study, a novel 3-D plane image input plant disease detection system was proposed and tested on orange fruits infected with black rot disease. This proposed system was conceptualized to have a revolving sample stand in order to be able to expose all the surface area of a sample under evaluation to the camera. However, the mechanical aspects being limited out at this stage of the study yet, the samples were turned manually during the testing phase. The proposed system achieved its objective by correctly classifying all the orange fruit samples (100% classification accuracy) presented to it, regardless of the black rot disease distribution on the surface area of the orange. Table 5.1 presents the classification results of a classical orange classifier (a classical model developed in Chapter 4) and that of a 3-Dimensional Image Input-Based Plant Disease Classification Model (proposed model) on orange samples with different black rot symptoms surface distribution.

Table 5-1: Comparative classification results of a classical orange classifier and a 3-Dimensional Image Input-Based Plant Disease Classification Model for oranges with different black rot symptoms surface distribution

Comparative classification results of a classical orange classifier and a 3-Dimensional Image Input-Based Plant Disease Classification Model for oranges with different black rot symptoms surface distribution

Classification Model	*Sample 1 (Healthy Orange)*	*Sample 2 (Black rot affected orange, Symptoms evenly all over the surface*	*Sample 2 (Black rot affected orange, Symptoms visible on a small area (5% of the total surface area)*
Classical Model Results	Healthy orange (99% confident)	Black rot affected orange (91% confident)	Healthy orange (100% confident)

Proposed Model Results	Healthy orange (97% confident)	Black rot affected orange (88% confident)	Black rot affected orange (100% confident)

Table 5.1 clearly demonstrates the added value of the proposed model relative to the traditional model. The proposed model can correctly classify the orange sample with an uneven distribution of surface black rot symptoms, while the traditional model cannot. Uneven symptom surface distribution in fruits may occur at the onset phase of the disease, hence the proposed model can detect the black rot disease earlier that the traditional model.

5.3. A Multi-Camera Image Input-Based Plant Disease Classification Model (Proposal 3)

This sub-section proposes yet another topology of a plant disease detection model that also aims at granting the classification model a 3-D 'view' similar to the one discussed in Section 5.4. This topology, however, achieves its objective by incorporating multiple input cameras arranged such that it captures the full surface of a sample plant part (especially spherical or cylindrical fruits) before a classification can be made. Classification of any sample plant part is based on a full view of such a sample, as opposed to a view of a certain section of the sample. The problem statement for this proposed model is similar to the one presented in Section 5.2.1. Therefore, the following sub-section conceptualizes this model based on multiple cameras.

5.3.1. Conceptualizing the proposed system

The proposed system operates as per the block diagram presented in Fig. 5.9. The first level has four (the number of cameras may vary depending on the specific design) image camera sensors to capture sample images at different angles such that the full surface area of a sample under evaluation is covered. The next level has a sequencer whose purpose is to select each image sensor to feed the input image into the deep learning classification algorithm sequentially from cameras 1 to 4. The classification model classifies each image from cameras 1 to 4 and stores the resulting classes into variables of output 1 to 4 respectively. The last operation compares the resulting outputs and makes a final classification using the following simple protocol (similar to Section 5.2.2):

- If at least 1 input image is classified as infected by a disease, set the final classification to a 'diseased sample'.
- If all the input images are classified as healthy, set the final classification to a 'healthy sample'.

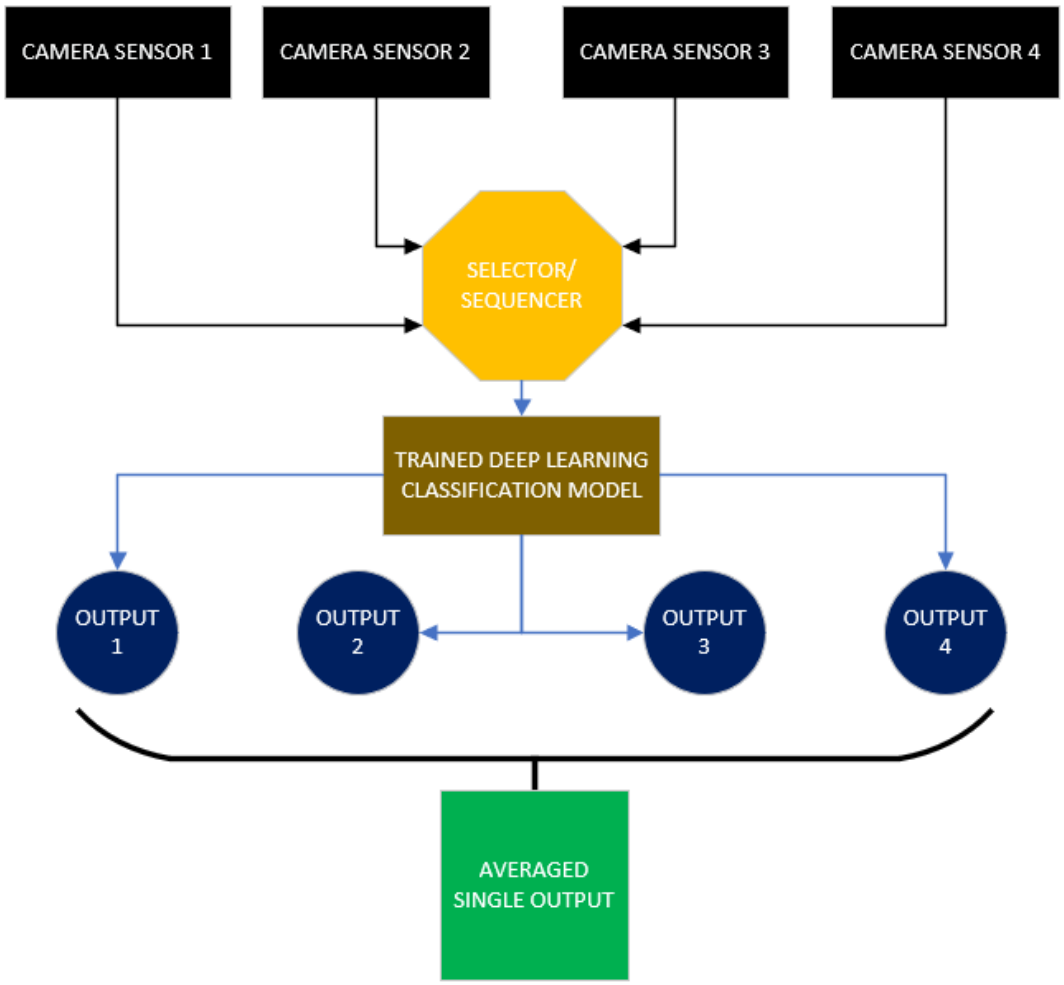


Figure 5-9: Architecture of a multi-camera-based plant disease detection model

For a 4-input camera model, the camera sensors should be arranged in a circular arrangement displaced 90˚ away from each other and equidistant from the position point of a sample under evaluation, as shown in Fig. 5.10. This arrangement is crucial to capture the images of a sample from all directions, effectively sending a 3-dimensional representation of a sample in a classification algorithm.

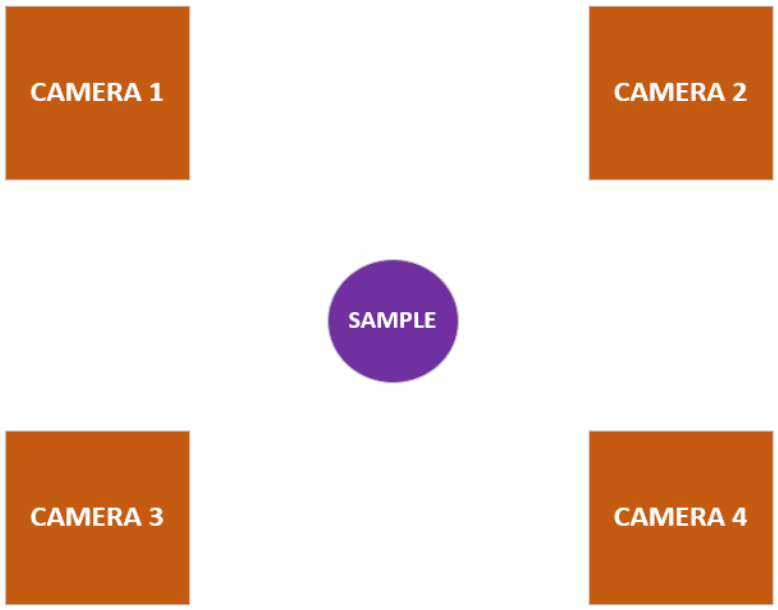


Figure 5-10: Multi-camera image infeed arrangement

5.3.2. Design of the Proposed Model

This sub-section develops a multi-camera image input-based plant disease classification model that distinguishes between healthy oranges and those that have black rot. This is also accomplished by using the MATLAB deep learning toolbox. The approach suggested in this proposal will also be illustrated using orange fruits and the black rot disease, as in Chapter 4. Therefore, the creation of a training dataset, training and validation of the proposed model will be similar to that of a classical orange classifier presented in Chapter 4, and hence will be omitted. However, the operation of the proposed system differs, and is therefore discussed in the next section.

5.3.3. Testing the Proposed Model

At this point, the design and validation of a neural network orange fruit classifier are complete. Now it is time to test this model on a practical example to see if it can classify correctly between healthy and black rot-affected oranges. A Lenovo ThinkPad was used as a host microprocessor to execute the training and testing proposed model. This computer has two USB ports and hence can support two UVC webcams. This was a major limitation encountered during the testing of this proposal. Hence, for demonstration purposes, two UVC webcams positioned 180˚ opposite each other on either side of a sample orange position were utilized for input images (refer to Fig. 5.11).

Figure 5-11: Input camera set-up for testing a trained healthy-black rot-infected orange classifier

Therefore, input images were obtained by acquiring sample images and passing them to an already-trained model. The test orange samples were collected from a local vendor where they had already been harvested. The oranges with the desired black-rot symptoms and disease distribution were specially requested from a vendor (similar to test samples in Section 5.2). The MATLAB code used to achieve the purpose of this section is shown in Appx. 9.

This program accepts snapshot images from the webcams (refer to Fig. 5.12) connected to a microprocessor of a host computer, Lenovo Laptop. These two infeed cameras capture the images of a sample orange on either side as shown in Fig. 5.11. Each of these two images passes through a previously trained classifier, is classified individually and the final classification results are formed by consolidating the individual results from webcams 1 and 2. If at least 1 camera captured an image classified as black rot-infected, the final result is set to a diseased orange. Otherwise, the orange is classified as healthy orange. Three different orange samples were collected to test the model operation, a healthy orange and two diseased oranges where black rot symptoms have spread evenly throughout the whole orange surface area and only on a small surface area on the other. The classification results of each sample are shown in Fig. 5.13-5.18.

```
Command Window
>> webcamlist

ans =

  2×1 cell array

    {'Integrated Camera'}
    {'Macally mZoomCam' }
```

Figure 5-12: Two input webcams for input images on a multiple camera infeed fruit disease classification model

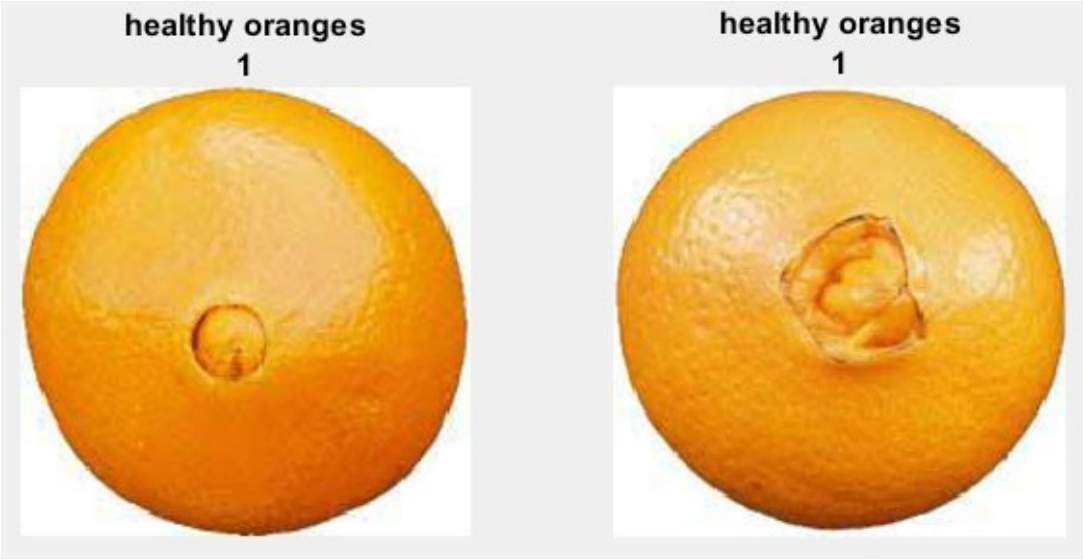


Figure 5-13: A healthy orange classified using two input cameras

```
Command Window
>> RealTimeFunction
Healthy Orange Detected!
Final Classification = Healthy orange!
```

Figure 5-14: Final classification for a healthy orange using two input cameras

Figure 5-15: A diseased orange with uneven black-rot distribution classification using two cameras

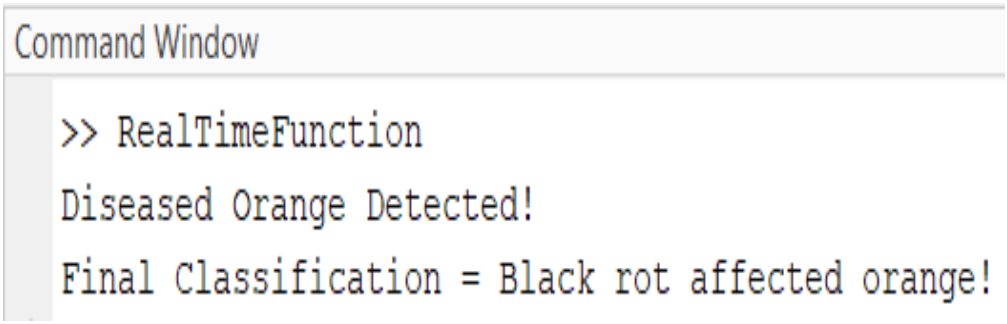


Figure 5-16: Final classification for a diseased orange with an uneven black-rot distribution using two input cameras

Figure 5-17: A diseased orange with an even black-rot distribution classification using two input cameras

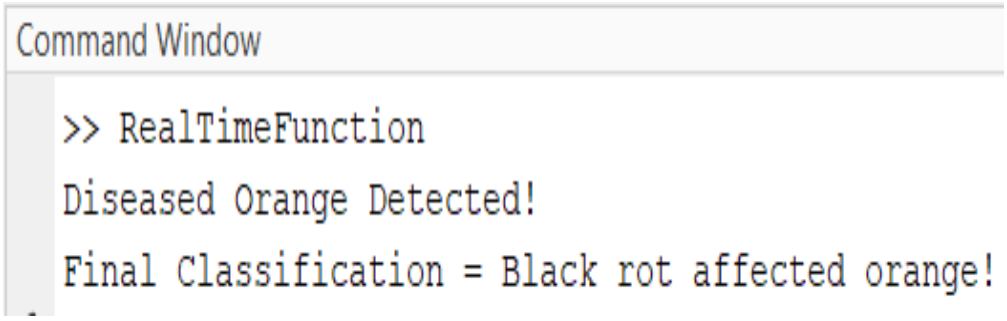


Figure 5-18: Final classification for a diseased orange with an even black-rot distribution using two input cameras

Fig. 5.13 shows two images of a healthy orange captured from two opposite angles by cameras 1 and 2, and each of them was classified correctly as a healthy orange. Fig. 5.14 shows the final classification results of this sample, which are derived from consolidating the results of each camera angle. Similarly, Fig. 5.15 shows two images of a diseased orange (with uneven black rot symptoms distribution) captured from two opposite angles by cameras 1 and 2, and each of them was classified correctly as a diseased orange and healthy orange. Fig. 5.16 shows the final classification results of this sample, which are also derived from consolidating the results of each camera angle. Lastly, Fig. 5.17 shows two images of a diseased orange (with even black rot symptoms distribution) captured from two opposite angles by cameras 1 and 2. Figure 5.18 shows the final classification results, which were formed similarly to those in Fig. 5.14 and 5.16. The classification accuracy of this multi-camera input fruit-disease classification model was 100%, and it has the added advantage of being capable of classifying correctly even the samples with an uneven disease distribution on their surface areas.

5.3.4. Evaluation and Comparison of a Multi-Camera Image Input-Based Plant Disease Classification Model Relative to a Healthy-Black Rot Affected Oranges Classification Model

In this study, a novel multi-camera input plant disease detection system was proposed and tested on orange fruits infected with black rot disease. This proposed system was conceptualized to have four camera inputs. However, due to a limitation of a host computer used as a microprocessor, the input cameras were scaled down to two. The proposed system achieved its objective by correctly classifying all the orange fruit samples presented to it, regardless of the black rot disease distribution on the surface area of the orange. All the methods (dataset collection, model training, model testing, etc.) were performed in accordance with the relevant guidelines and regulations. Table 5.2 presents the classification results of a classical orange classifier (a classical model developed in Chapter 4) and that of a Multi-Camera Image Input-Based Plant Disease Classification Model (proposed model) on orange samples with different black rot symptoms surface distribution.

Table 5-2: Comparative classification results of a classical orange classifier and a Multi-Camera Image Input-Based Plant Disease Classification Model for oranges with different black rot symptoms surface distribution

Comparative classification results of a classical orange classifier and a Multi-Camera Image Input-Based Plant Disease Classification Model for oranges with different black rot symptoms surface distribution			
Classification Model	*Sample 1 (Healthy Orange)*	*Sample 2 (Black rot affected orange, Symptoms evenly all over the surface*	*Sample 2 (Black rot affected orange, Symptoms visible on a small area (5% of the total surface area)*
Classical Model Results	Healthy orange (96% confident)	Black rot affected orange (95% confident)	Healthy orange (86% confident)
Proposed Model Results	Healthy orange (93% confident)	Black rot affected orange (94% confident)	Black rot affected orange (99% confident)

Again, Table 5.2 clearly demonstrates the added capability of the proposed model relative to the traditional model. The proposed model can also correctly classify the orange sample with an uneven distribution of surface black rot symptoms while the traditional model cannot. Since uneven symptom surface distribution in fruits generally occurs at the onset phase of the disease, the proposed model can thus also detect the black rot disease earlier than the traditional model.

5.4. A Hybrid Fruit Disease-Quality Monitoring and Sorting Model (H-DQMS Model, Proposal 4)

This sub-section proposes a hybrid deep learning model that combines systems that are originally stand-alone into a single universal system. This system assumes a commercial farm that produces different fruit breeds and different fruit species altogether. For this specific study, the researcher assumes a farm that specializes in orange and apple production. The proposed

system, a Hybrid Fruit Disease-Quality Monitoring and Sorting Model (H-DQMS Model), can differentiate between different species of fruit (in this case, different species are oranges and apples); differentiate between different breeds of the same fruit species (in this case different breed are golden apples and red apples); differentiate between a good and bad quality fruit, either a healthy orange, red or golden apple with or without defects (such as punchers, peeled skin or damaged fruits); and lastly, the system is able to sort them according to their different categories.

5.4.1. Problem Statement

The majority of farms in developing regions such as Africa still rely on manual harvesting methods, as seen in Fig. 5.1 (a). This harvesting method is faced with many challenges, ranging from its risky nature to low effectiveness and efficiency. Harvesters must rely on their knowledge background and experience to select good-quality fruits during harvesting. This often opens big room for error and translates to high revenue losses in farms and poor-quality end-products such as fruit juices during manufacturing if poor-quality fruits make it to trade. As a result, farmers must hire more workforce for long durations to manually sort fruit according to different slots to try and get the quality aspect correct, refer to Fig. 5.1(b). This too is not without its uncertainties and errors as sorters rely on their naked eye to judge between good and bad quality fruits. Even in well-established farms where precision agricultural harvesting tools are utilized, manual sorting is still utilized to some extent.

(a) (b)

Figure 5-19:(a) Manual orange harvesting; (b) Manual orange sorting

Poor warehousing may lead to the mixing of different fruit breeds and species. Mixing can also occur during manual harvesting due to human error, or during automated harvesting due to fruit being locked up in machinery moving parts, or due to different fruit fields too close to each other. Mixing different fruits/foods unintentionally is unacceptable and can be deemed food contamination. Food contamination might undermine consumers' health. A good example of this in the case of fruits and fruit products is as follows: Assume a consumer that is suffering from severe or chronic ulcers who has been given a list of foods they should not consume by their physician such as tomatoes, oranges, lemons, etc. If this consumer then opts to go for an apple juice and surprisingly finds that this fruit juice has traces of oranges that accidentally passed through a production line during this apple juice's production and made it through to the end-product, then this consumer may be offended.

Fruit may also be damaged during harvesting. This may occur due to fruits interacting with different tools utilized during manual (refer to Fig. 5.2 (a-b)) harvesting and due to machinery moving parts during automated harvesting. Fruits may also be damaged by birds or different types of pests. Manual harvesting might discriminate such damaged fruits to some extent, but automated harvesting might not do so at all. This then necessitates extensive sorting operations after harvesting before fruits can be dispatched to the market. Above all, some production companies might require a specific grade of fruits from a farm to suit their production needs and that grade may range across a spectrum of parameters such as weight, colour and overall quality aspects.

(a) (b)

Figure 5-20.: (a) Plastic fruit picking tool; (b) Metal fruit picking tool

Hence, this section concludes that the importance of a model such as an H-DQMSM cannot be over-emphasized in today's farming operations. The following section briefly outlines the key objectives of an H-DQMS Model and afterwards, the actual model is designed.

5.4.2. Key Objectives of an H-DQMS Model

The prime objectives of this model are as follows:

- To maximize an orange and apple farm yield by minimizing losses during harvesting operations;
- To inform timely corrective actions by quickly identifying diseases on fruits;
- To improve the time turnover of the harvesting operations by replacing slow and erroneous manual sorting labour with a fast and accurate automated sorting system;
- To improve a farm's revenues by saving on time and losses; and
- To improve and maintain good supplier-customer relationships by ensuring that only top-quality fruits are sent to customers as per their expectations, hence drastically reducing disputes.

5.4.3. Conceptualizing the Proposed Model

The design of an H-DQMS model with follow the same design procedure as that of a healthy-black-rot affected classification system in Chapter 4. Fig. 5.21 shows a block diagram of an H-DQMS model. This system has an infeed conveyor system, the main Hybrid-DQMS control system, and the distribution conveyor system of different classes of fruits. The infeed conveyor system has a width of 5 lines and it reduces in steps of 1 line until it reaches a single line width. This reducing section of the infeed conveyor is called a pressure-free combiner (PFC). Its purpose is to reduce an influx of fruits (oranges or apples or a combination of the two) that comes in soldier pattern into a single profile pattern.

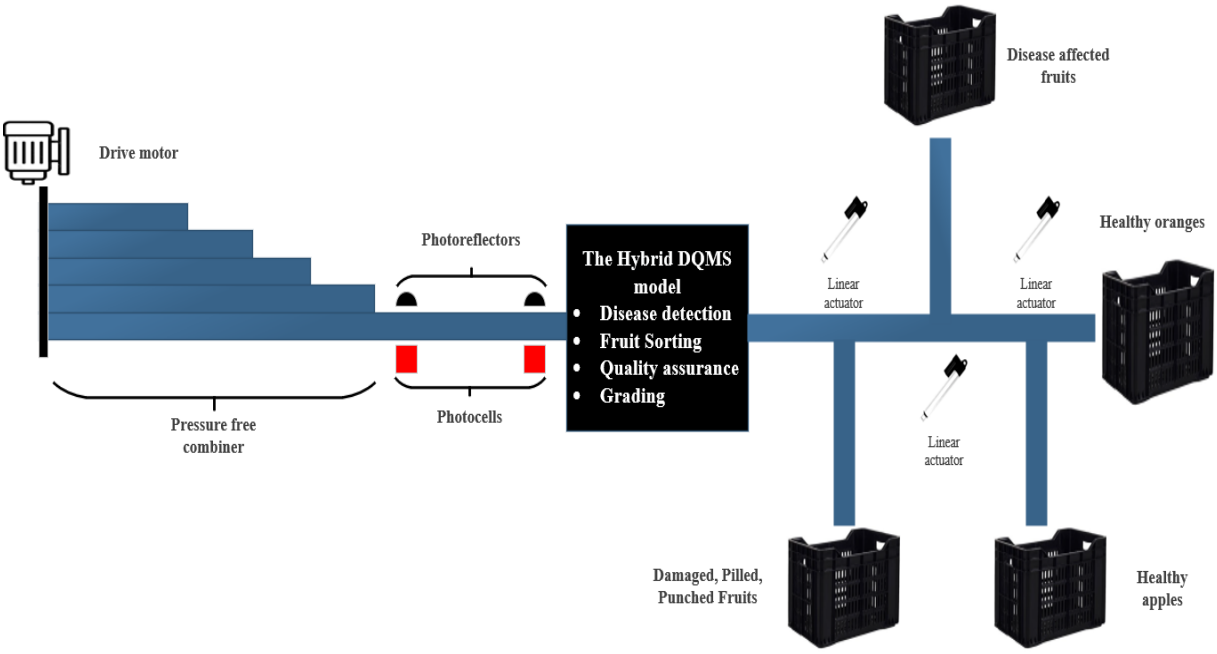


Figure 5-21: The proposed Hybrid-DQMS model

(a) **(b)**

Figure 5-22: (a) Oranges in soldier pattern; (b) Oranges in a single profile

The soldier pattern occurs when fruit run downstream in a conveyor arranged in a side-by-side pattern, which resembles that of soldiers in a parade, refer to Fig. 5.22 (a). On the other hand, a single profile occurs when the fruits run downstream in a conveyor arranged in a back-to-back pattern, forming a single line as shown in Fig. 5.22 (b). At a single profile section in

the infeed conveyor, a set of photocells and photo reflectors (and sometimes pressure cells) are monitoring and controlling the infeed to the Hybrid-DQMS control circuit. If all the photocells are made (the laser beam between a photocell and a reflector is blocked and there is no reflected light back to the photocell) and the pressure sensors are made (The conveyor is full of fruit), a signal will be sent to the drive motor to slow down the conveyor system and eventually bring it to halt. Otherwise, if the photocell and pressure cells are not made, this implies that there are gaps in the incoming stream of fruits and hence the signal is sent to the drive motor to ramp up the infeed conveyor speed and close the gaps. After a Hybrid-DQMS control circuit, there is a distribution set of a conveyor system that is mounted with mechanical actuators. During operation, a Hybrid-DQMS control circuit captures the image of an incoming fruit, classifies it, and sends a signal to activate the actuator corresponding to a correct class of fruit being evaluated at a particular time. This control circuit utilizes encoders and servomotors to track the exact position of a fruit under evaluation across the distribution conveyor and knocks it to a bin corresponding to its class.

This research study is only limited to designing a proposed Hybrid-DQMS model and will omit all the mechanical aspects of this system for future work consideration. Therefore, the next section looks at a Hybrid-DQMS model in more detail.

5.4.4. Conceptualizing the Proposed System Control Circuit

Fig. 5.23 shows a simplified Hybrid-DQMS control circuit that can be seen in Fig. 5.21, hence it is a 'zoom-in' of the black box shown in Fig. 5.21 labelled the 'Hybrid-DQMS model'. This control circuit is made up of three main sections, viz., the change parts, the main system, and the output sections. The change-parts section comprises hardware and soft items that can be used interchangeably in this system depending on its primary function at a given time. The change parts are the sensors by which the control circuit acquires input stimuli from its environment of operation, and different MATLAB program functions are used to perform different classification tasks. In the case of this research study, the change parts are the camera system of a webcam to capture a live-stream video of fruits as they pass through the control circuit; the fruit presence sensors to prime the control circuit to a state of high alert while anticipating an incoming fruit; load cell in the case when a control circuit is performing fruit grading in terms of fruit weights; knock-off mechanical actuators in the distribution conveyor system; and the MATLAB program instructing the control circuit.

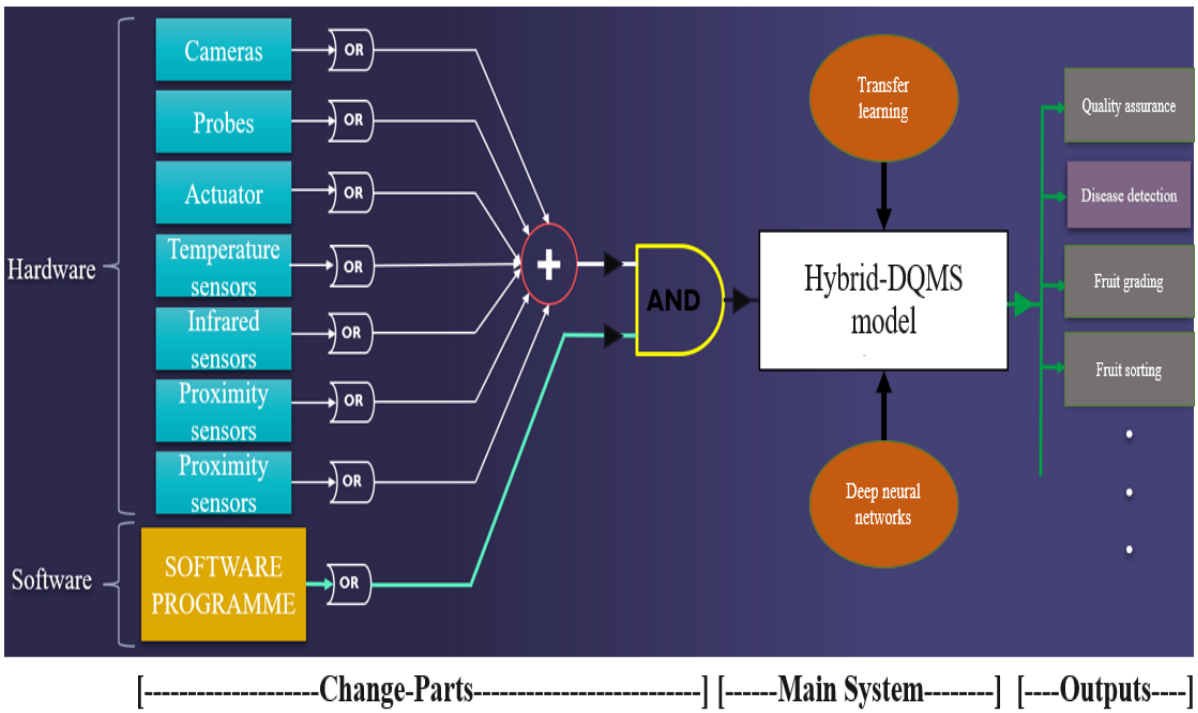


Figure 5-23: A Hybrid-DQMS control circuit block diagram

The main system comprises the central processing unit (CPU), which is a microprocessor that receives instructions from the MATLAB program. The microprocessor can be a stand-alone (e.g. raspberry pi) or a host computer such as a laptop or PC. This is the section where the training and validation of a deep neural network occur. The classification and subsequent sorting action are controlled in this part of the control circuit. A deep neural network held in this section is trained through transfer learning. The output section produces different output signals depending on the decision made in the main system, and the corresponding actuator is activated at the correct time stipulated by the encoders and/or servo motors.

The following section deals with the actual design of a Hybrid-DQMS control circuit. This design follows the same procedure as the one employed during the design of a healthy-black-rot affected oranges classification system.

5.4.5. Modelling and testing of a Hybrid-DQMS control circuit

This section designs a deep learning classifier that classifies healthy oranges and apples, botch-affected and black rot affect apples and oranges respectively, as well as damaged apples and oranges. This is also achieved through the utilization of the MATLAB deep learning toolbox. This section will only cover the training and validation of the classifier, while the next

section covers the testing of a classification. Again, the dataset has to be developed and divided into training and validation datasets, similar to the case of an orange classifier in Chapter 4, The only difference in this case is that the dataset is categorized into 6 classes as opposed to the two classes in Chapter 4.

- **Creating a dataset for a Hybrid-DQMS model**

The training and validating datasets were also sourced from Kaggle. The dataset folder has six sub-folders in it called 'healthy orange', 'healthy apple', 'black rot affected orange', botch affected apple', 'damaged orange', and 'damaged apple'. Each folder contains 100 corresponding images. This dataset is then loaded into MATLAB's current folder, as shown in Fig. 5.24, before proceeding to the next step. Although the training and validation image dataset were acquired from Kaggle, during the operation of this Hybrid-DQMS control circuit when training and validation are complete, the model will be getting input images through snapshots from a live video feed from a Macally webcam in Fig. 4.4.

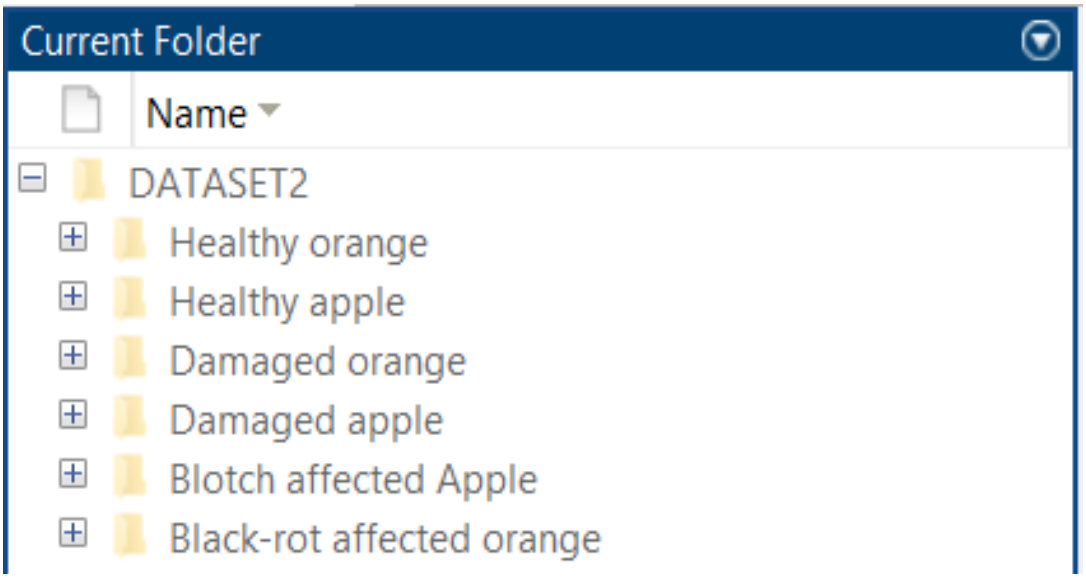


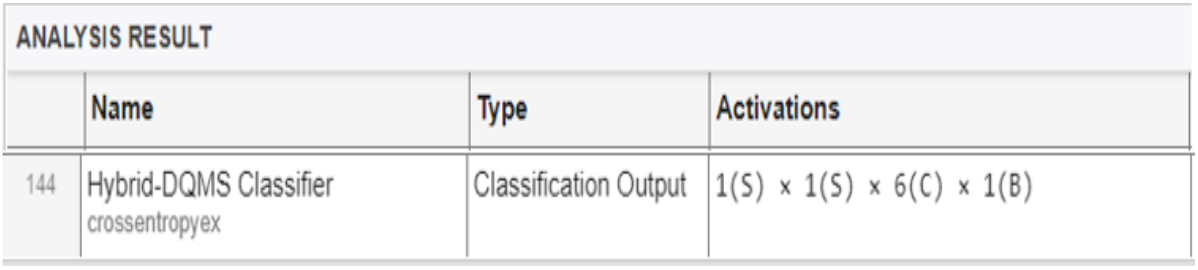
Figure 5-24: A training and validation dataset for training a Hybrid-DQMS model

- Training Google-Net to form a Hybrid-DQMS model

The next step is training a neural network and the training procedure resembles the one presented in Section 4.3.1. The MATLAB program that was used to achieve the training and validation of a Hybrid-DQMS model is shown in Appx. A10. This code is similar to the one explained in section 4.3.1, except that it now trains a neural network in 6 different classes and

the last neural network layer has been re-named 'Hybrid-DQMS Classifier' to match the primary function of this proposed improved network.

Fig. 5.25 shows the new classification layer of a Hybrid-DQMS model and one can notice in the activation column that the classifier now has 6 possible output classes, hence the number of classes in the training dataset. Comparing Fig. 5.25 to Fig. 4.19, the healthy-black rot-affected orange classifier had 2 possible output classes, while the Hybrid-DQMS model has six.


ANALYSIS RESULT

	Name	Type	Activations
144	Hybrid-DQMS Classifier crossentropyex	Classification Output	1(S) × 1(S) × 6(C) × 1(B)

Figure 5-25: The classification layer of a Hybrid-DQMS model

The training progress was also observed during the training of a Hybrid-DQMS model and it can be viewed in Fig. 5.26. The training was set to occur over 10 epochs, with 5 iterations each. By the 4th epoch, the training efficiency had reached 100% while the training losses reached 0%. The validation accuracy increased steadily from the 1st to the 10th epoch, while the validation losses decreased accordingly. The average validation accuracy at the end of the training and validation accuracy and training duration of a Hybrid-DQMS model was 82.33% and 291 seconds respectively, as shown in Fig. 5.27.

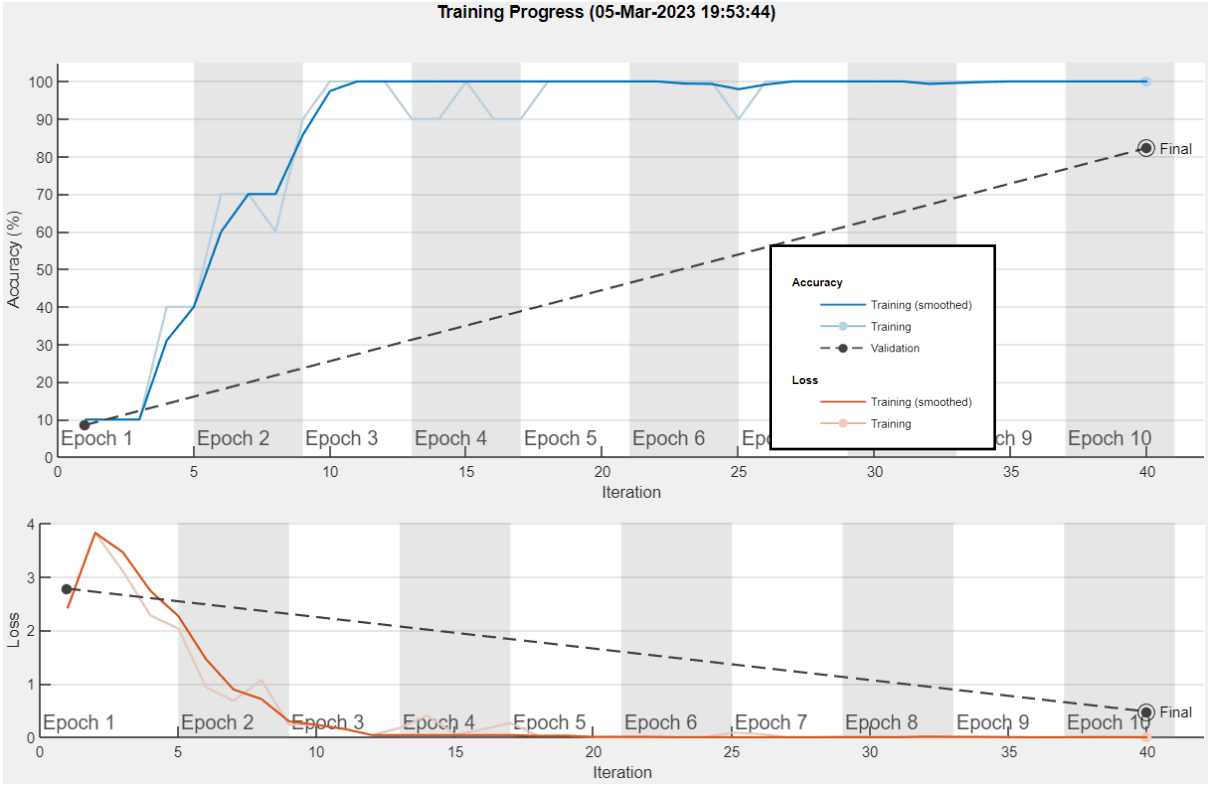


Figure 5-26: Training progress of a Hybrid-DQMS model

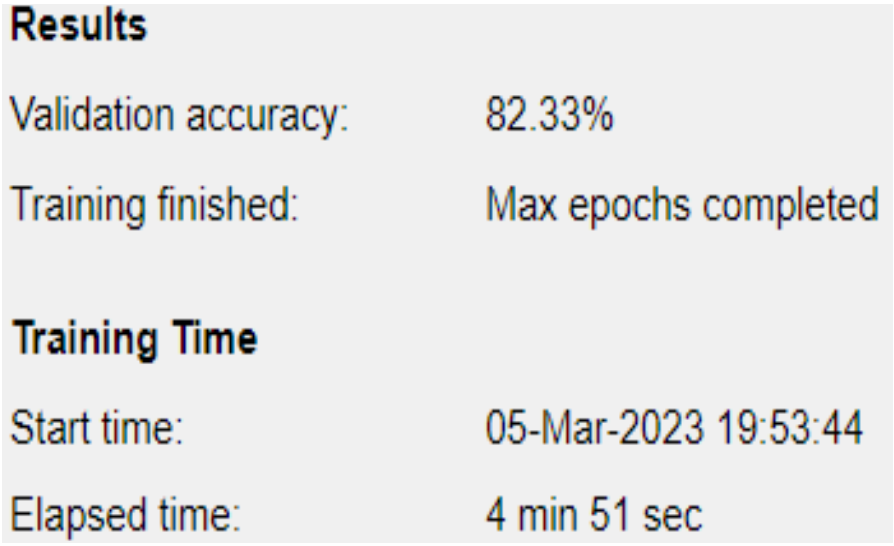


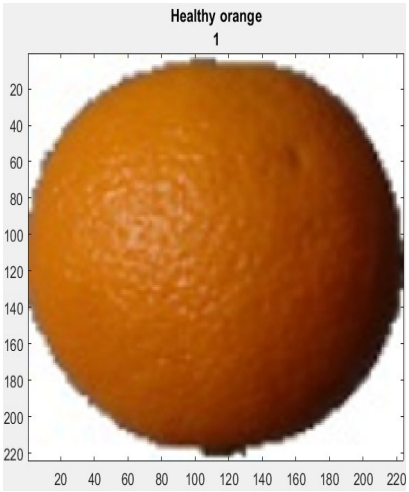
Figure 5-27: Validation efficiency and training duration of a Hybrid-DQMS model

- **Testing a Hybrid- DQMS Model for a practical application**

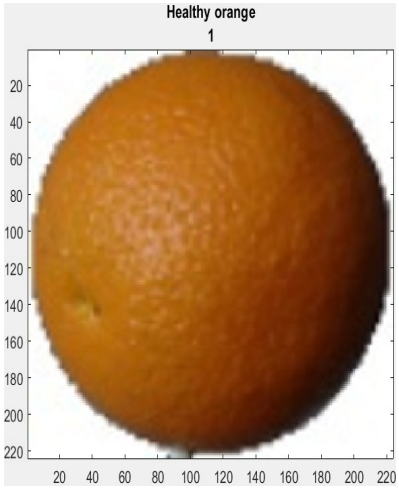
At this point, a Hybrid-DQMS model is ready for action and this section serves the purpose of testing whether the model is operating as it is supposed to. This will be achieved by acquiring a sample image corresponding to all six classes and passing it to a trained Hybrid-DQMS model

and observing the classification results. The MATLAB code used to achieve the purpose of this section is shown in Appx. A.11.

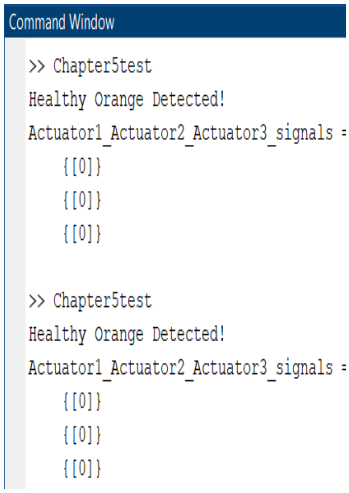
This MATLAB program is separated into two main sections, a Hybrid-DQMS model testing and actuators' clock signal generations. The model testing section is similar to the one presented in Section 4.2.3.2. It takes snapshots from a Macally webcam live feed video, re-sizes the images into 224×224 pixels, and classifies these images according to their classes using an already trained Hybrid-DQMS model. The classified images are then printed in image windows with their predicted class and probability of that prediction. However, since there are no appropriate sample fruits at the time of documentation of this thesis, 12 test images (2 per class) were downloaded from Google open sources to be utilized for testing. The actuator clock signal section is creating clock signals to activate the corresponding actuator each time a certain class of fruit passes a Hybrid-DQMS control circuit. The testing results are shown in Fig. 5.28.



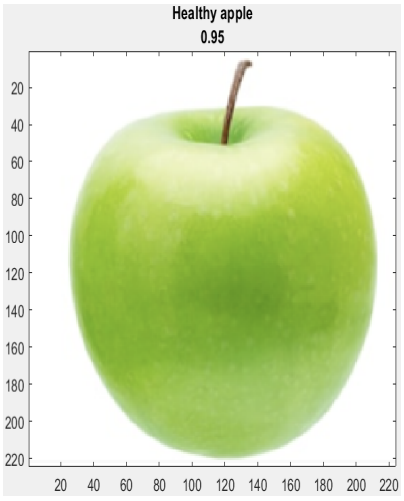
(a)



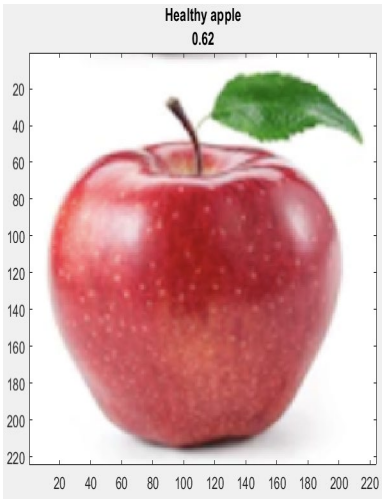
(b)



(c)



(d)

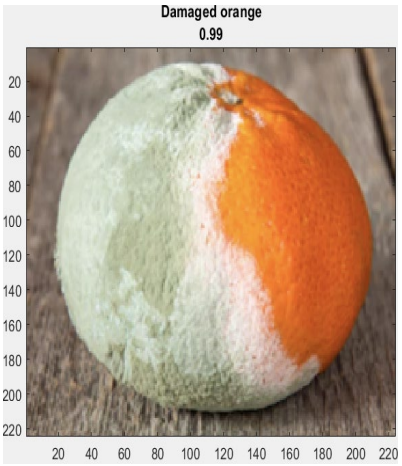


(e)

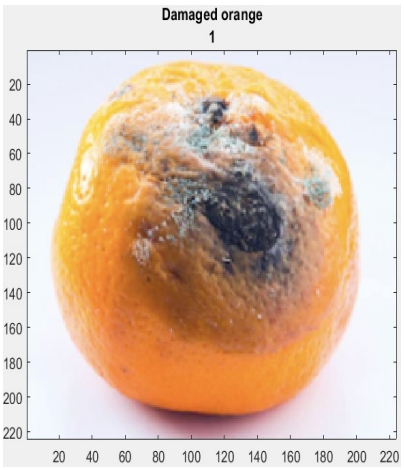
```
Command Window
>> Chapter5test
Healthy Apple Detected!
Actuator1_Actuator2_Actuator3_signals =
    {[0]}
    {[0]}
    {[1]}

>> Chapter5test
Healthy Apple Detected!
Actuator1_Actuator2_Actuator3_signals =
    {[0]}
    {[0]}
    {[1]}
```

(f)



(g)

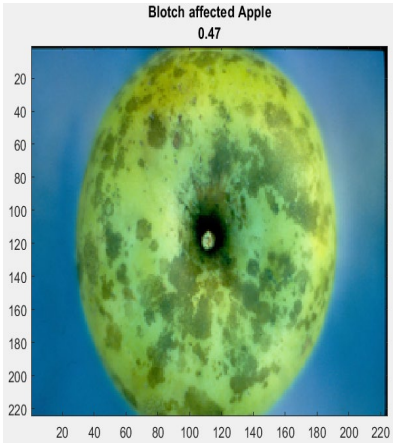


(h)

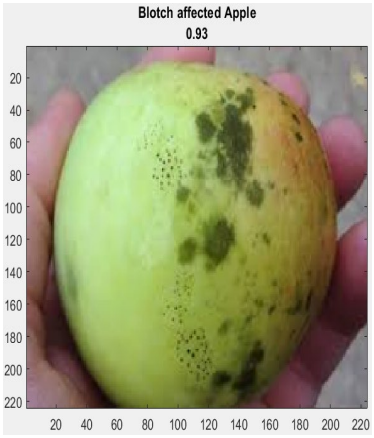
```
Command Window
>> Chapter5test
Damaged fruit Detected!
Actuator1_Actuator2_Actuator3_signals
    {[1]}
    {[0]}
    {[0]}

>> Chapter5test
Damaged fruit Detected!
Actuator1_Actuator2_Actuator3_signals
    {[1]}
    {[0]}
    {[0]}
```

(i)



(j)

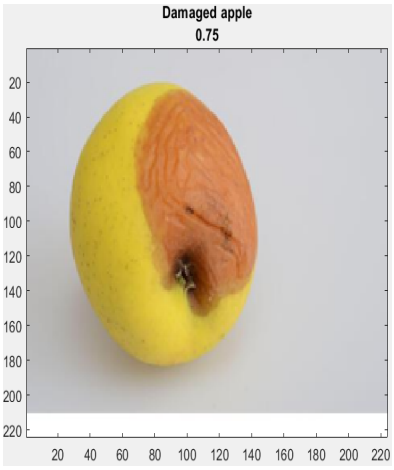


(k)

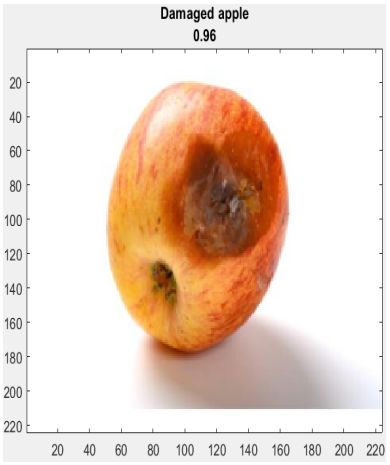
```
Command Window
>> Chapter5test
Diseased Fruit Detected!
Actuator1_Actuator2_Actuator3_signals =
    {[0]}
    {[1]}
    {[0]}

>> Chapter5test
Diseased Fruit Detected!
Actuator1_Actuator2_Actuator3_signals =
    {[0]}
    {[1]}
    {[0]}
```

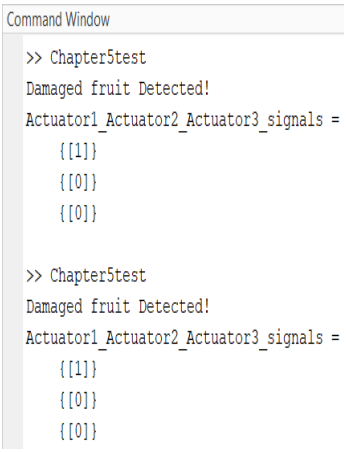
(l)



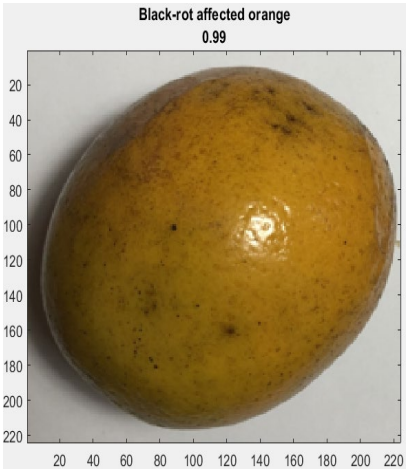
(m)



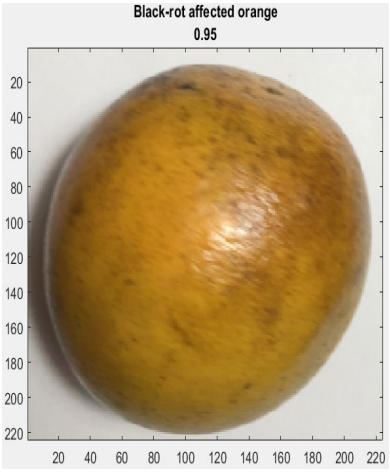
(n)



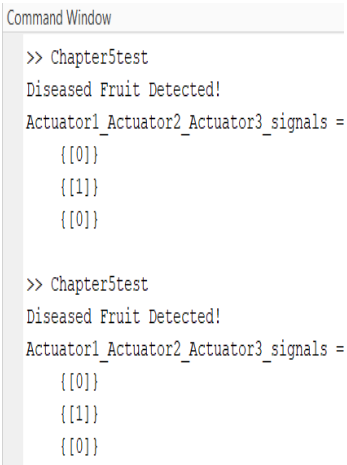
(o)



(p)



(q)



(r)

Figure 5-28: (a) 1st healthy orange classification; (b) 2nd healthy orange classification; (c) Actuator clock healthy oranges classification; (d) 1st healthy apple classification; (e) 2nd healthy apple classification; (f) signals during healthy apple classification (g) 1st damaged orange classification; (h) 2nd damaged orange (i) Actuator clock signals during damaged oranges classification; (j)1st botch-affected apple classificatio affected apple classification; (l) Actuator clock signals during botch affected apple classification; (m) 1st classification; (n) 2nd damaged apple classification; (o) Actuator clock signals during damaged apple cla 1st black rot affected orange classification; (q) 2nd black rot affected orange classification; (r) Actuato during black rot affected oranges classification

5.4.6. Evaluation and comparison of a Hybrid- DQMS model relative to a healthy-black rot affected oranges classification model

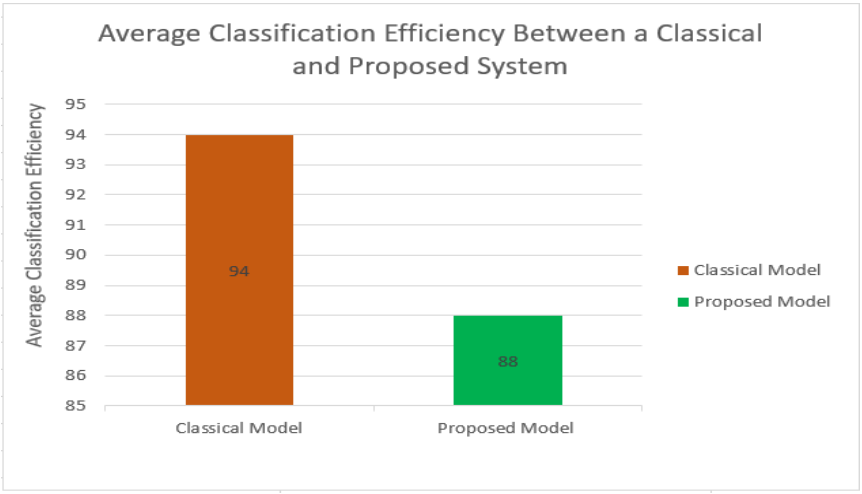
This section evaluates the success of a Hybrid-DQMS model by comparing it to a classical model, a healthy-Black rot-affected model. Fig. 5.28 (a-r) shows that a Hybrid-DQMS model

successfully classified healthy, diseased and damaged fruits (apples and oranges) with 100% accuracy. Over and above, a Hybrid-DQMS model correctly generated an actuator clock signal to re-direct a fruit to its corresponding container. Table 5.1 compares a healthy-black rot-affected orange classical detection model to a Hybrid-DQMS model.

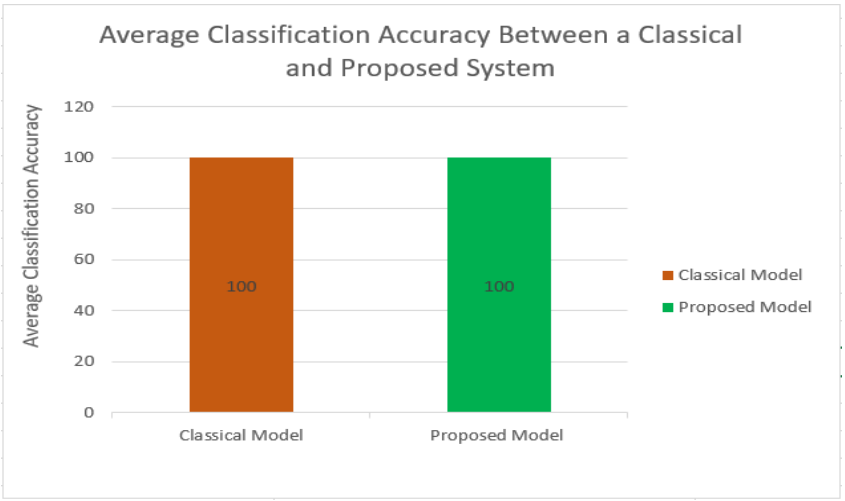
Table 5-3: Comparison of a healthy-black rot-affected orange classical detection model to a Hybrid-DQMS model

Conceptual and operational comparison of an orange classical detection model and a Hybrid-DQMS model	
healthy-black rot affected the orange classifier	*Hybrid- DQMS model*
• Single operation system	• Multiple operations single system
• Only performs the disease classification	• Performs multiple disease applications, quality assurance, and sorting and grading
• Performs monitoring only	• Performance monitoring and the corresponding is executed: Discard damaged and diseased fruits and keep the healthy fruits
• Fixed: Operation on one fruit species	• Flexible and allows for future development: Few change-part may be required to add a new capability.
• Does not contribute to the harvesting process	• Makes harvesting faster by eliminating slow and erroneous manual fruit sorting and grading.

Table 5.3 outlined the fundamental conceptual and operational distinctions between a healthy-black rot-affected orange classical detection model and a Hybrid-DQMS model. The validation accuracy of a healthy-black rot-affected orange classical detection model was 98.58%, whilst that of a Hybrid-DQMS model was 82.33%. It is lower for a Hybrid-DQMS model partly because of a lower number of training and validation images per class which is 100 images (versus 160 images for a classical healthy-black rot-affected orange classifier). Fig. 5.11 shows the graphical key performance parameters of a healthy-black rot-affected orange classical detection model relative to a Hybrid-DQMS model. Fig. 5.29 (a) shows the average classification efficiency between the two models, as already discussed.



(a)



(b)

Figure 5-29: (a) Average classification efficiency; (b) Average classification accuracy of an orange classifier versus a Hybrid-DQMS model

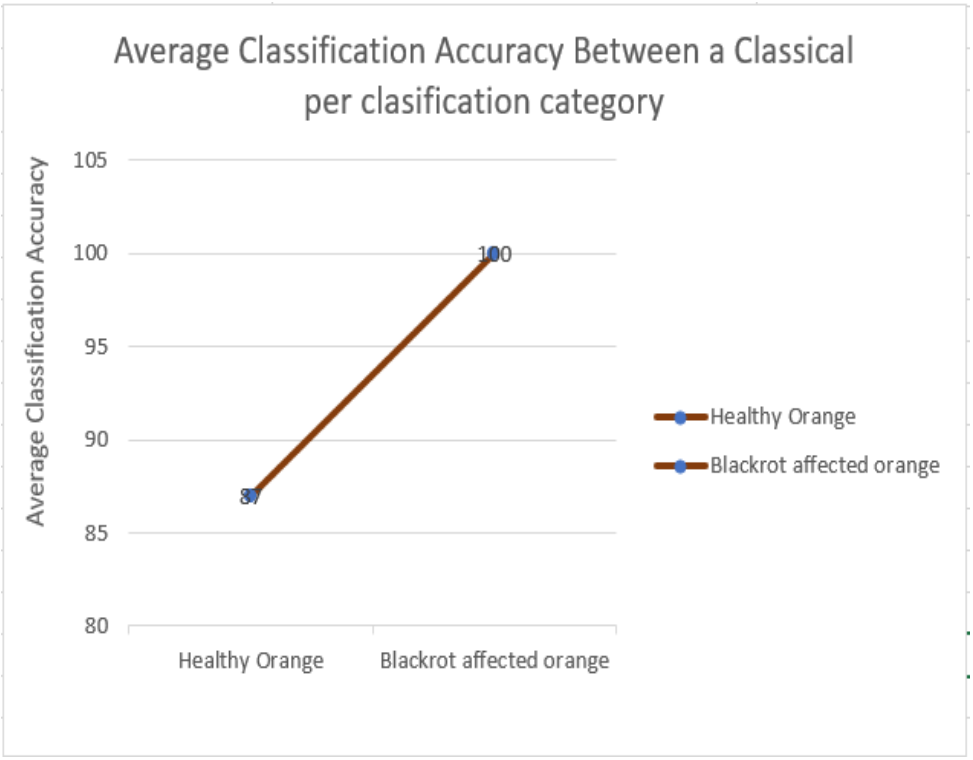


Figure 5-30: Classification Efficiency of a healthy-black rot affected orange classical detection model per classification category

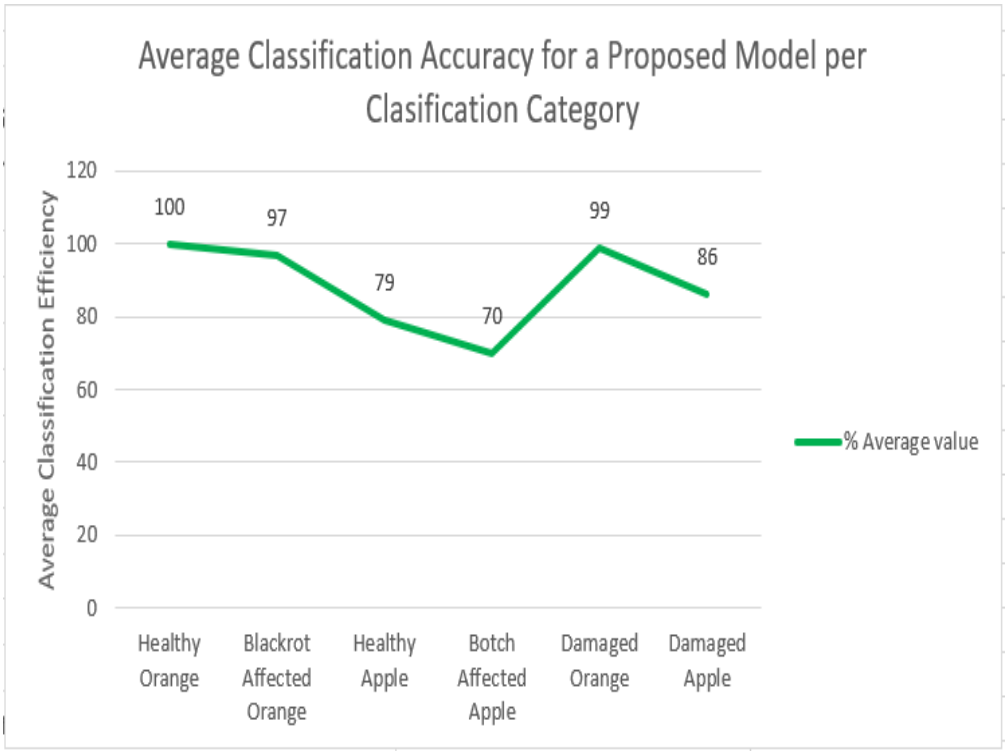


Figure 5-31: Classification Efficiency of a Hybrid- DQMS Model per classification category

Fig. 5.29 (b) shows the average classification accuracy, which is 100% for both systems since no test image was wrongly classified. Fig. 5.30 and 5.31 show the classification efficiency for a healthy-black rot-affected orange classical detection model and a Hybrid-DQMS model respectively, across different fruit categories with a minimum and maximum of 70% and 100% respectively.

5.5. Conclusion

This chapter proposed three different classification models: a multi-camera and a revolving sample stand models aimed at granting a classification model a '3-Dimensional view' of the sample (for a more efficient and accurate classification), as well as a Hybrid-DQMS model which performs several more functions other than classifying a disease in oranges, as seen from a healthy-black rot-affected orange classical detection presented in a traditional system in Chapter 4. This H-DQMS model can also perform quality assurance by identifying fruits with any form of damage (pest, bird damages, bruises, punches, etc); perform fruit grading by identifying healthy fruits deemed fit for trade; and perform the overall sorting operations. The H-DQMS model was capable of correctly classifying 100% of the test fruits passed through it and its classification confidence interval varies from category to category. Increasing the amount of training and validation data and improving the variety of sample images within the training and validation dataset, can be utilized to improve the efficiency and confidence interval of an H-DQMS model.

CHAPTER 6: CONCLUSION

6.1. The overall conclusion of the study

The purpose of this study was to investigate plant disease detection models to identify opportunities for future research and make proposals that seek to improve the functionality of the traditional models. This study and all its related experiments have been performed mainly on oranges, and apples to a less extent.

The motive of this study has been the raging food insecurity globally and the crucial economical role played by the agricultural sector in the developing regions of the world, such as on the Continent of Africa. According to the literature, crop diseases are one of the leading causes of yield losses on farms. Scholars have increasingly focused on plant disease detection models over the last few decades, proposing several different topologies, and achieving great results. However, this study has, through an extensive literature review of the open access related publications, managed to notice several research opportunities existing in this field of study. The identified opportunities were the following:

- Little or no literature discussed the real-time monitoring of the onset signs of diseases before they spread throughout the whole plant part.
- Few publications discussed real-time monitoring and real-time mitigation measures such as actuation operations, spraying pesticides and spraying fertilizers, to name a few examples.
- Very little research discusses the combination of these monitoring and phenotyping tasks into 1 system to reduce costs and improve technology availability to farmers and add convenience.
- Most studies utilize single image sensors to capture the input images to be processed by a classification model. This may be problematic, especially in cases where spherical or cylindrical specimens are used for classification as some parts may be hidden from the camera's line of sight.
- Few studies focus on the post-harvesting benefits of these classification models.

- No study was encountered during the literature review discussing the relationship between the classification efficiency and accuracy of these classification models to the lighting conditions in which they operate.
- Fewer studies discuss the role that can potentially be played by these plant disease classification models in high throughput and high-speed production plants.

Therefore, this study has been built to take advantage of a few of the research opportunities listed above. A classical plant disease classification model was designed and modelled to classify healthy oranges from black rot-affected oranges (in Chapter 4). This classification model was developed to serve as a control from which the proposed systems in the study will be referenced. During the modelling process of this reference classification model, another opportunity was identified in image pre-processing techniques that are used to prepare the sample images for feature extraction and subsequent classification. This study has noticed that dissolving or dissociating an RGB image into its original red, green, and blue colour planes and then performing thresholding in either of the plane(s) that offers the best colour-intensity contrast between the foreground and the background yields better feature segmentation. Therefore, an improved image pre-processing technique has been the first proposal of this study using the earlier-mentioned premise.

The second and third proposals of this study were inspired by one of the earlier-mentioned research opportunities concerning the use of single cameras for capturing input images. Most fruits are spherical (such as oranges, apples, etc.), while others are cylindrical (such as bananas, sugar cane, etc.) A single-input camera capturing a single-input image cannot 'see' the full surface area of either a spherical or cylindrical fruit. Fruit disease symptoms have been observed not to spread evenly across the surface area of the fruit, as are the fruit defects such as punchers, pilled skin, and damage from pests or fruit birds. This implies that the images sent to the classification model for evaluation might be missing crucial details due to the orientation of the fruit when the input image was captured. This shortcoming impacts the classification accuracy

and hence the effectiveness of the traditional plant/fruit disease detection models. Therefore, in order to mitigate this problem, this study has made two key proposals.

The second proposal of this study was a novel plant disease detection model that utilizes multiple input camera sensors to capture the full surface of a spherical or cylindrical sample, hence the 'multi-camera image input plant disease detection model'. The third proposal was a classification model which, conversely, utilizes a single image sensor to capture input images but with the capability to turn the sample around its axes while capturing multiple images that cover the full surface area of a spherical or cylindrical fruit sample. Both the third and fourth proposals of this study seek to grant a classification model a '3-D view' of a sample under evaluation to ensure that the classification model processes its full surface area before making a final classification. These proposals were tested and displayed a classification accuracy of 100%.

The fourth and last proposals of this study was concerned with combining several crucial processes in a farming operation (such as disease classification, grading, quality assurance, etc.) that are originally stand-alone systems into a single universal hybrid system capable of performing these multiple tasks. The purpose of this proposal is to lower the cost of this kind of technology and hence improve its availability and accessibility, even in low-budget farms. This led to the concept of a hybrid fruit disease-quality monitoring and sorting model. This model can perform plant disease classification and sorting operations based on different grades such as diseases, damages and overall quality. This model was also conceptualized, modelled and tested in this study. Its operation was rated 100% as it classified and sorted all the samples given to it accurately.

Several proposals have been made in this study and they are promising in concept. The major shortcoming of this study is that most of the proposals made are electromechanical and would have needed some mechanical fabrication to be fully tested under practical conditions. This study was limited to testing the electrical part of these proposals and the mechanical aspects were excluded because the contribution of this study is more inclined towards the electrical

aspects of the proposals made in this research study. Therefore, the first and second proposals were fully tested while the third and last were partially tested.

6.2. Future Work Recommendations

Several future work recommendations can be made from this study. The first one would be to complete the mechanical designs of the proposed 3-D image input plant disease classification model and the H-DQMS model to observe the performance of these systems under practical settings. It is also important to understand the impact of lighting conditions on the classification efficiency and accuracy of these plant disease detection models. Can these models operate at night? What benefit can infrared image-sensing technology bring to these systems? What role can space technology play in these plant disease detection systems? What would be the impact on the classification results (accuracy and efficiency) if different ML classification algorithms such as SMV, k-NN, Fuzzy logic, etc. were utilized instead of deep learning?

REFERENCES

[1] A. Badage, “Crop disease detection using machine learning: Indian agriculture,” *Int. Res. J. Eng. Technol,* vol. 5, no. 9, pp. 866-869, 2018.

[2] U. Ukaegbu, L. Tartibu, T. Laseinde, M. Okwu, and I. Olayode, "A deep learning algorithm for detection of potassium deficiency in a red grapevine and spraying actuation using a raspberry pi3." pp. 1-6.

[3] U. Shruthi, V. Nagaveni, and B. Raghavendra, "A review on machine learning classification techniques for plant disease detection." pp. 281-284.

[4] S. Adekar, and A. Raul, “Detection of Plant Leaf Diseases using Machine Learning,” *2019 International Research Journal of Engineering and Technology (IRJET)*, 2019.

[5] R. S. Pachade, “A REVIEW ON COMPARATIVE STUDY OF MODERN TECHNIQUES TO ENHANCE INDIAN FARMING USING AI AND MACHINE LEARNING.”

[6] M.-L. du Preez, “4IR and Water Smart Agriculture in Southern Africa: A Watch List of Key Technological Advances,” 2020.

[7] M. S. Hoosain, B. S. Paul, and S. Ramakrishna, “The impact of 4IR digital technologies and circular thinking on the United Nations sustainable development goals,” *Sustainability,* vol. 12, no. 23, pp. 10143, 2020.

[8] B. Swaminathan, “Identification of Plant Disease and Wetness/Dryness Detection.”

[9] M. Islam, K. A. Wahid, A. V. Dinh, and P. Bhowmik, “Model of dehydration and assessment of moisture content on onion using EIS,” *Journal of food science and technology,* vol. 56, no. 6, pp. 2814-2824, 2019.

[10] S. Anju, B. Chaitra, C. Roopashree, K. Lathashree, and S. Gowtham, “Various Approaches for Plant Disease Detection,” 2021.

[11] S. Swain, S. K. Nayak, and S. S. Barik, “A review on plant leaf diseases detection and classification based on machine learning models,” *Mukt shabd,* vol. 9, no. 6, pp. 5195-5205, 2020.

[12] N. Prashar, "A Review on Plant Disease Detection Techniques." pp. 501-506.

[13] N. Agrawal, J. Singhai, and D. K. Agarwal, "Grape leaf disease detection and classification using multi-class support vector machine." pp. 238-244.

[14] Z. A. Dar, S. A. Dar, J. A. Khan, A. A. Lone, S. Langyan, B. Lone, R. Kanth, A. Iqbal, J. Rane, and S. H. Wani, "Identification for surrogate drought tolerance in maize inbred lines utilizing high-throughput phenomics approach," *Plos one,* vol. 16, no. 7, pp. e0254318, 2021.

[15] F. Perez-Sanz, P. J. Navarro, and M. Egea-Cortines, "Plant phenomics: An overview of image acquisition technologies and image data analysis algorithms," *GigaScience,* vol. 6, no. 11, pp. gix092, 2017.

[16] K. Padmavathi, and K. Thangadurai, "Implementation of RGB and grayscale images in plant leaves disease detection–comparative study," *Indian Journal of Science and Technology,* vol. 9, no. 6, pp. 1-6, 2016.

[17] A. Kern, "CCD and CMOS detectors," 2019.

[18] S. S. Magazov, "Image recovery on defective pixels of a CMOS and CCD arrays," *Informatsionnye Tekhnologii i Vychslitel'nye Sistemy*, no. 3, pp. 25-40, 2019.

[19] D. Defrianto, M. Shiddiq, U. Malik, V. Asyana, and Y. Soerbakti, "Fluorescence spectrum analysis on leaf and fruit using the ImageJ software application," *Science, Technology & Communication Journal,* vol. 3, no. 1, pp. 1-6, 2022.

[20] A. F. A. Netto, R. N. Martins, G. S. A. De Souza, F. F. L. Dos Santos, and J. T. F. Rosas, "Evaluation of a low-cost camera for agricultural applications," *J. Exp. Agric. Int,* vol. 32, pp. 1-9, 2019.

[21] P. K. R. Maddikunta, S. Hakak, M. Alazab, S. Bhattacharya, T. R. Gadekallu, W. Z. Khan, and Q.-V. Pham, "Unmanned aerial vehicles in smart agriculture: Applications, requirements, and challenges," *IEEE Sensors Journal,* vol. 21, no. 16, pp. 17608-17619, 2021.

[22] J. Trivedi, Y. Shamnani, and R. Gajjar, "Plant leaf disease detection using machine learning." pp. 267-276.

[23] H. Wang, S. Shang, D. Wang, X. He, K. Feng, and H. Zhu, “Plant disease detection and classification method based on the optimized lightweight YOLOv5 model,” *Agriculture,* vol. 12, no. 7, pp. 931, 2022.

[24] P. B. Wakhare, S. Neduncheliyan, and K. R. Thakur, "Study of Disease Identification in Pomegranate Using Leaf Detection Technique." pp. 1-6.

[25] B. K. Ekka, and B. S. Behera, “Disease Detection in Plant Leaf Using Image Processing Technique,” *International Journal of Progressive Research in Science and Engineering,* vol. 1, no. 4, pp. 151-155, 2020.

[26] N. R. Kolhalkar, and V. Krishnan, “Mechatronics system for diagnosis and treatment of major diseases in grape vineyards based on image processing,” *Materials Today: Proceedings,* vol. 23, pp. 549-556, 2020.

[27] K. Takada, “A Study on Learning Algorithms of Value and Policy Functions in Hex,” 北海道大学, 2019.

[28] X. Contreras, N. Amberg, A. Davaatseren, A. H. Hansen, J. Sonntag, L. Andersen, T. Bernthaler, C. Streicher, A. Heger, and R. L. Johnson, “A genome-wide library of MADM mice for single-cell genetic mosaic analysis,” *Cell reports,* vol. 35, no. 12, pp. 109274, 2021.

[29] C. Mazur, J. Ayers, J. Humphrey, G. Hains, and Y. Khmelevsky, "Machine Learning Prediction of Gamer’s Private Networks (GPN® S)." pp. 107-123.

[30] V. Vijayalakshmi, and K. Venkatachalapathy, “Comparison of predicting student’s performance using machine learning algorithms,” *International Journal of Intelligent Systems and Applications,* vol. 11, no. 12, pp. 34, 2019.

[31] K. S. Adewole, A. G. Akintola, S. A. Salihu, N. Faruk, and R. G. Jimoh, "Hybrid rule-based model for phishing URLs detection." pp. 119-135.

[32] D. Krivoguz, "Validation of landslide susceptibility map using ROCR package in R." pp. 188-192.

[33] M. Sieber, S. Klar, M. F. Vassiliou, and I. Anastasopoulos, “Robustness of simplified analysis methods for rocking structures on compliant soil,” *Earthquake Engineering & Structural Dynamics,* vol. 49, no. 14, pp. 1388-1405, 2020.

[34] C. Aybar, Q. Wu, L. Bautista, R. Yali, and A. Barja, “rgee: An R package for interacting with Google Earth Engine,” *Journal of Open Source Software,* vol. 5, no. 51, pp. 2272, 2020.

[35] M. Schweinberger, "Tree-based models in R," The University of Queensland. Retrieved from https://slcla dal. github. io …, 2021.

[36] S. Pölsterl, “scikit-survival: A Library for Time-to-Event Analysis Built on Top of scikit-learn,” *J. Mach. Learn. Res.,* vol. 21, no. 212, pp. 1-6, 2020.

[37] N. Melnykova, R. Kulievych, Y. Vycluk, K. Melnykova, and V. Melnykov, “Anomalies Detecting in Medical Metrics Using Machine Learning Tools,” *Procedia Computer Science,* vol. 198, pp. 718-723, 2022.

[38] E. J. Gómez-Hernández, P. A. Martínez, B. Peccerillo, S. Bartolini, J. M. García, and G. Bernabé, “Using PHAST to port Caffe library: First experiences and lessons learned,” *arXiv preprint arXiv:2005.13076*, 2020.

[39] M. N. Gevorkyan, A. V. Demidova, T. S. Demidova, and A. A. Sobolev, “Review and comparative analysis of machine learning libraries for machine learning,” *Discrete and Continuous Models and Applied Computational Science,* vol. 27, no. 4, pp. 305-315, 2019.

[40] M. Weber, H. Wang, S. Qiao, J. Xie, M. D. Collins, Y. Zhu, L. Yuan, D. Kim, Q. Yu, and D. Cremers, “Deeplab2: A tensorflow library for deep labeling,” *arXiv preprint arXiv:2106.09748*, 2021.

[41] G. DESHPANDE, “Deceptive security using Python.”

[42] V. Patel, “Open Source Frameworks for Deep Learning and Machine Learning.”

[43] A. Pocock, “Tribuo: Machine Learning with Provenance in Java,” *arXiv preprint arXiv:2110.03022*, 2021.

[44] E. Schubert, and A. Zimek, “ELKI: A large open-source library for data analysis-ELKI Release 0.7. 5" Heidelberg",” *arXiv preprint arXiv:1902.03616*, 2019.

[45] Y. Liu, X. S. Shan, W. Kan, B. Kong, H. Li, C. Wang, T. Yang, C. Li, Y. Tan, and H. Qu, "JSAT."

[46] A. Bhatia, and B. Kaluza, *Machine Learning in Java: Helpful techniques to design, build, and deploy powerful machine learning applications in Java*: Packt Publishing Ltd, 2018.

[47] H. Luu, *Beginning Apache Spark 2: with resilient distributed datasets, Spark SQL, structured streaming and Spark machine learning library*: Apress, 2018.

[48] M. K. Vanam, B. A. Jiwani, A. Swathi, and V. Madhavi, "High-performance machine learning and data science-based implementation using Weka," *Materials Today: Proceedings*, 2021.

[49] T. Saha, N. Aaraj, N. Ajjarapu, and N. K. Jha, "SHARKS: Smart Hacking Approaches for RisK Scanning in Internet-of-Things and cyber-physical systems based on machine learning," *IEEE Transactions on Emerging Topics in Computing*, 2021.

[50] R. R. Curtin, M. Edel, M. Lozhnikov, Y. Mentekidis, S. Ghaisas, and S. Zhang, "mlpack 3: a fast, flexible machine learning library," *Journal of Open Source Software,* vol. 3, no. 26, pp. 726, 2018.

[51] Z. Wen, J. Shi, Q. Li, B. He, and J. Chen, "ThunderSVM: A fast SVM library on GPUs and CPUs," *The Journal of Machine Learning Research,* vol. 19, no. 1, pp. 797-801, 2018.

[52] K. Kolodiazhnyi, *Hands-On Machine Learning with C++: Build, train, and deploy end-to-end machine learning and deep learning pipelines*: Packt Publishing Ltd, 2020.

[53] A. Mohan, A. K. Singh, B. Kumar, and R. Dwivedi, "Review on remote sensing methods for landslide detection using machine and deep learning," *Transactions on Emerging Telecommunications Technologies,* vol. 32, no. 7, pp. e3998, 2021.

[54] R. Prasad, and V. Rohokale, "Artificial intelligence and machine learning in cyber security," *Cyber Security: The Lifeline of Information and Communication Technology*, pp. 231-247: Springer, 2020.

[55] F. Garcia-Lamont, J. Cervantes, A. López, and L. Rodriguez, "Segmentation of images by color features: A survey," *Neurocomputing,* vol. 292, pp. 1-27, 2018.

[56] A. Wang, W. Zhang, and X. Wei, “A review on weed detection using ground-based machine vision and image processing techniques,” *Computers and electronics in agriculture,* vol. 158, pp. 226-240, 2019.

[57] J. Ker, S. P. Singh, Y. Bai, J. Rao, T. Lim, and L. Wang, “Image thresholding improves 3-dimensional convolutional neural network diagnosis of different acute brain hemorrhages on computed tomography scans,” *Sensors,* vol. 19, no. 9, pp. 2167, 2019.

[58] A. Kumar, and A. Tiwari, “A comparative study of otsu thresholding and k-means algorithm of image segmentation,” *Int. J. Eng. Technol. Res,* vol. 9, pp. 2454-4698, 2019.

[59] L. Zhang, L. Zou, C. Wu, J. Jia, and J. Chen, “Method of famous tea sprout identification and segmentation based on an improved watershed algorithm,” *Computers and Electronics in Agriculture,* vol. 184, pp. 106108, 2021.

[60] L. Xie, J. Qi, L. Pan, and S. Wali, “Integrating deep convolutional neural networks with marker-controlled watershed for overlapping nuclei segmentation in histopathology images,” *Neurocomputing,* vol. 376, pp. 166-179, 2020.

[61] P. M. Anger, L. Prechtl, M. Elsner, R. Niessner, and N. P. Ivleva, “Implementation of an open source algorithm for particle recognition and morphological characterization for microplastic analysis utilizing Raman microspectroscopy,” *Analytical Methods,* vol. 11, no. 27, pp. 3483-3489, 2019.

[62] S. Jadhav, and B. Garg, "Comparative Analysis of Image Segmentation Techniques for Real Field Crop Images." pp. 1-17.

[63] C. Li, X. Zhao, and H. Ru, "GrabCut Algorithm Fusion of Extreme Point Features." pp. 33-38.

[64] J. F. Randrianasoa, C. Kurtz, E. Desjardin, and N. Passat, “AGAT: Building and evaluating binary partition trees for image segmentation,” *SoftwareX,* vol. 16, pp. 100855, 2021.

[65] N. Zhu, X. Liu, Z. Liu, K. Hu, Y. Wang, J. Tan, M. Huang, Q. Zhu, X. Ji, and Y. Jiang, “Deep learning for smart agriculture: Concepts, tools, applications, and opportunities,” *International Journal of Agricultural and Biological Engineering,* vol. 11, no. 4, pp. 32-44, 2018.

[66] Q. Zhang, Y. Liu, C. Gong, Y. Chen, and H. Yu, “Applications of deep learning for dense scenes analysis in agriculture: A review,” *Sensors,* vol. 20, no. 5, pp. 1520, 2020.

[67] D. Ireri, E. Belal, C. Okinda, N. Makange, and C. Ji, “A computer vision system for defect discrimination and grading in tomatoes using machine learning and image processing,” *Artificial Intelligence in Agriculture,* vol. 2, pp. 28-37, 2019.

[68] S. Singh[1], and P. P. Kaur, “A study of geometric features extraction from plant leafs,” 2019.

[69] M. Martsepp, T. Laas, K. Laas, J. Priimets, S. Tõkke, and V. Mikli, “Dependence of multifractal analysis parameters on the darkness of a processed image,” *Chaos, Solitons & Fractals,* vol. 156, pp. 111811, 2022.

[70] J. M. Ponce, A. Aquino, and J. M. Andújar, “Olive-fruit variety classification by means of image processing and convolutional neural networks,” *IEEE Access,* vol. 7, pp. 147629-147641, 2019.

[71] N. R. Bhimte, and V. Thool, "Diseases detection of cotton leaf spot using image processing and SVM classifier." pp. 340-344.

[72] G. K. Sandhu, and R. Kaur, "Plant disease detection techniques: a review." pp. 34-38.

[73] P. Alagumariappan, N. J. Dewan, G. N. Muthukrishnan, B. K. B. Raju, R. A. A. Bilal, and V. Sankaran, “Intelligent plant disease identification system using Machine Learning,” *Engineering Proceedings,* vol. 2, no. 1, pp. 49, 2020.

[74] A. A. Bharate, and M. Shirdhonkar, "Classification of grape leaves using KNN and SVM classifiers." pp. 745-749.

[75] S. Sivasakthi, and M. MCA, "Plant leaf disease identification using image processing and svm, ann classifier methods." pp. 30-31.

[76] C. U. Kumari, S. J. Prasad, and G. Mounika, "Leaf disease detection: feature extraction with K-means clustering and classification with ANN." pp. 1095-1098.

[77] R. Azadnia, and K. Kheiralipour, “Recognition of leaves of different medicinal plant species using a robust image processing algorithm and artificial neural networks classifier,” *Journal of Applied Research on Medicinal and Aromatic Plants,* vol. 25, pp. 100327, 2021.

[78] M. Vaishnnave, K. S. Devi, P. Srinivasan, and G. A. P. Jothi, "Detection and classification of groundnut leaf diseases using KNN classifier." pp. 1-5.

[79] E. Hossain, M. F. Hossain, and M. A. Rahaman, "A color and texture based approach for the detection and classification of plant leaf disease using KNN classifier." pp. 1-6.

[80] J. Singh, and H. Kaur, "Plant disease detection based on region-based segmentation and KNN classifier." pp. 1667-1675.

[81] A. Bakhshipour, and H. Zareiforoush, "Development of a fuzzy model for differentiating peanut plant from broadleaf weeds using image features," *Plant methods,* vol. 16, no. 1, pp. 1-16, 2020.

[82] H. Sabrol, and S. Kumar, "Plant leaf disease detection using adaptive neuro-fuzzy classification." pp. 434-443.

[83] P. Sutha, A. Nandhu Kishore, V. Jayanthi, A. Periyanan, and P. Vahima, "Plant Disease Detection Using Fuzzy Classification," *Annals of the Romanian Society for Cell Biology*, pp. 9430-9441, 2021.

[84] K. P. Panigrahi, H. Das, A. K. Sahoo, and S. C. Moharana, "Maize leaf disease detection and classification using machine learning algorithms," *Progress in Computing, Analytics and Networking*, pp. 659-669: Springer, 2020.

[85] Y. Kashyap, and S. Shrivastava, "ROLE OF VARIOUS SEGMENTATION AND CLASSIFICATION ALGORITHMS IN PLANT DISEASE DETECTION."

[86] A. Dwari, A. Tarasia, A. Jena, S. Sarkar, S. K. Jena, and S. Sahoo, "Smart Solution for Leaf Disease and Crop Health Detection," *Advances in Intelligent Computing and Communication*, pp. 231-241: Springer, 2021.

[87] N. K. KUMAR, and A. PRATHYUSHA, "SOLANUM TUBEROSUM LEAF DISEASES DETECTION USING DEEP LEARNING."

[88] S. Hossain, R. M. Mou, M. M. Hasan, S. Chakraborty, and M. A. Razzak, "Recognition and detection of tea leaf's diseases using support vector machine." pp. 150-154.

[89] S. Coulibaly, B. Kamsu-Foguem, D. Kamissoko, and D. Traore, “Deep neural networks with transfer learning in millet crop images,” *Computers in Industry,* vol. 108, pp. 115-120, 2019.

[90] N. Cherukuri, G. R. Kumar, O. Gandhi, V. S. K. Thotakura, D. NagaMani, and C. Z. Basha, "Automated Classification of rice leaf disease using Deep Learning Approach." pp. 1206-1210.

[91] E. Khalili, S. Kouchaki, S. Ramazi, and F. Ghanati, “Machine learning techniques for soybean charcoal rot disease prediction,” *Frontiers in plant science,* vol. 11, pp. 590529, 2020.

[92] M. P. Patil, “Nearest neighborhood approach for identification of cucumber mosaic virus infection and yellow sigatoka disease on banana plants,” 2022.

[93] H. Orchi, M. Sadik, and M. Khaldoun, “On using artificial intelligence and the internet of things for crop disease detection: A contemporary survey,” *Agriculture,* vol. 12, no. 1, pp. 9, 2021.

[94] D. Zhang, X. Zhou, J. Zhang, Y. Lan, C. Xu, and D. Liang, “Detection of rice sheath blight using an unmanned aerial system with high-resolution color and multispectral imaging,” *PloS one,* vol. 13, no. 5, pp. e0187470, 2018.

[95] G. Yashodha, and D. Shalini, “An integrated approach for predicting and broadcasting tea leaf disease at early stage using IoT with machine learning–a review,” *Materials Today: Proceedings,* vol. 37, pp. 484-488, 2021.

[96] A. V. Zubler, and J.-Y. Yoon, “Proximal methods for plant stress detection using optical sensors and machine learning,” *Biosensors,* vol. 10, no. 12, pp. 193, 2020.

[97] B. L. Thushara, and T. M. Rasool, “Analysis of plant diseases using expectation maximization detection with BP-ANN classification,” *XIII (VIII)*, 2020.

[98] P. S. Thakur, P. Khanna, T. Sheorey, and A. Ojha, “Trends in vision-based machine learning techniques for plant disease identification: A systematic review,” *Expert Systems with Applications*, pp. 118117, 2022.

[99] A. I. Khan, S. Quadri, and S. Banday, “Deep learning for apple diseases: classification and identification,” *International Journal of Computational Intelligence Studies,* vol. 10, no. 1, pp. 1-12, 2021.

[100] A. J. Das, R. Ravinath, T. Usha, B. S. Rohith, H. Ekambaram, M. K. Prasannakumar, N. Ramesh, and S. K. Middha, “Microbiome Analysis of the Rhizosphere from Wilt Infected Pomegranate Reveals Complex Adaptations in Fusarium—A Preliminary Study,” *Agriculture,* vol. 11, no. 9, pp. 831, 2021.

[101] S. S. Gaikwad, “Fungi classification using convolution neural network,” *Turkish Journal of Computer and Mathematics Education (TURCOMAT),* vol. 12, no. 10, pp. 4563-4569, 2021.

[102] D. Priya, “Cotton leaf disease detection using Faster R-CNN with Region Proposal Network,” *International Journal of Biology and Biomedicine,* vol. 6, 2021.

[103] B. M. Joshi, and H. Bhavsar, “Plant leaf disease detection and control: A survey,” *Journal of Information and Optimization Sciences,* vol. 41, no. 2, pp. 475-487, 2020.

[104] G. Gangadevi, and C. Jayakumar, "Review of Classifiers Used for Identification and Classification of Plant Leaf Diseases." pp. 445-459.

[105] V. Vučić, M. Grabež, A. Trchounian, and A. Arsić, “Composition and potential health benefits of pomegranate: a review,” *Current pharmaceutical design,* vol. 25, no. 16, pp. 1817-1827, 2019.

[106] V. Sahni, S. Srivastava, and R. Khan, “Modelling techniques to improve the quality of food using artificial intelligence,” *Journal of Food Quality,* vol. 2021, 2021.

[107] S. Patidar, A. Pandey, B. A. Shirish, and A. Sriram, "Rice plant disease detection and classification using deep residual learning." pp. 278-293.

[108] M. Sharif, M. A. Khan, Z. Iqbal, M. F. Azam, M. I. U. Lali, and M. Y. Javed, “Detection and classification of citrus diseases in agriculture based on optimized weighted segmentation and feature selection,” *Computers and electronics in agriculture,* vol. 150, pp. 220-234, 2018.

[109] T. Hayit, H. Erbay, F. Varçın, F. Hayit, and N. Akci, “Determination of the severity level of yellow rust disease in wheat by using convolutional neural networks,” *Journal of Plant Pathology,* vol. 103, no. 3, pp. 923-934, 2021.

[110] S. S. Jasim, and A. A. M. Al-Taei, “A Comparison Between SVM and K-NN for classification of Plant Diseases,” *Diyala Journal for Pure Science,* vol. 14, no. 2, pp. 94-105, 2018.

[111] P. DAYANG, and A. S. K. MELI, “AS: Evaluation of image segmentation algorithms for plant disease detection,” *Int. J. Image Graph. Signal Process.(IJIGSP),* vol. 13, no. 5, pp. 14-26, 2021.

[112] M. Agarwal, S. K. Gupta, and K. Biswas, “Development of Efficient CNN model for Tomato crop disease identification,” *Sustainable Computing: Informatics and Systems,* vol. 28, pp. 100407, 2020.

[113] R. D. Devi, S. A. Nandhini, R. Hemalatha, and S. Radha, "IoT enabled efficient detection and classification of plant diseases for agricultural applications." pp. 447-451.

[114] S. S. Harakannanavar, J. M. Rudagi, V. I. Puranikmath, A. Siddiqua, and R. Pramodhini, “Plant Leaf Disease Detection using Computer Vision and Machine Learning Algorithms,” *Global Transitions Proceedings*, 2022.

[115] H. Altıparmak, M. Al Shahadat, E. Kiani, and K. Dimililer, "Fuzzy classification for strawberry diseases-infection using machine vision and soft-computing techniques." pp. 429-436.

[116] M. Toseef, and M. J. Khan, “An intelligent mobile application for diagnosis of crop diseases in Pakistan using fuzzy inference system,” *Computers and Electronics in Agriculture,* vol. 153, pp. 1-11, 2018.

[117] Mahlein, Anne-Katrin. "Plant disease detection by imaging sensors–parallels and specific demands for precision agriculture and plant phenotyping." *Plant disease* 100.2 (2016): 241-251.

[118] Ashwinkumar, S., et al. "Automated plant leaf disease detection and classification using optimal MobileNet based convolutional neural networks." Materials Today: Proceedings 51 (2022): 480-487.

[119] Buja, Ilaria, et al. "Advances in plant disease detection and monitoring: From traditional assays to in-field diagnostics." *Sensors* 21.6 (2021): 2129.

[120] Rehman, Zia Ur, et al. "Recognizing apple leaf diseases using a novel parallel real-time processing framework based on MASK RCNN and transfer learning: An application for smart agriculture." *IET Image Processing* 15.10 (2021): 2157-2168.

[121] Zia Ur Rehman, Muhammad, et al. "Classification of citrus plant diseases using deep transfer learning." *Computers, Materials & Continua* 70.1 (2021).

[122] Zia Ur Rehman, Muhammad, et al. "Classification of citrus plant diseases using deep transfer learning." *Computers, Materials & Continua* 70.1 (2021).

[123] Chen, Junde, Defu Zhang, and Yaser Ahangari Nanehkaran. "Identifying plant diseases using deep transfer learning and enhanced lightweight network." Multimedia tools and applications 79 (2020): 31497-31515.

APPENDICES

A1. MATLAB Code for Loading Images into Workspace and Resizing Them into 226×226 Pixels

```
Healthy_Orange = imread("Healthy Orange.jfif");
Diseased_Orange = imread("Diseased Orange.jfif");
Pest_Orange = imread("Pest Orange.jpg");
Resized_Healthy_Orange = imresize(Healthy_Orange, [226 226]);
Resized_Diseased_Orange = imresize(Diseased_Orange, [226 226]);
Resized_Pest_Orange = imresize(Pest_Orange, [226 226]);
```

A2. MATLAB Code for Generation Grayscale Images of RGB Images Shown in Fig. 4.5

```
Healthy_Orange_GreyScale = im2gray(Resized_Healthy_Orange);
Diseased_Orange_GreyScale = im2gray(Resized_Diseased_Orange);
Pest_Orange_Greyscale = im2gray(Resized_Pest_Orange);
```

A3. MATLAB Code for Dissolving the RGB Images in Fig. 4.5 into R, G, and B Planes

```
R_Healthy_Orange = Healthy_Orange (:,:,1);
R_Diseased_Orange = Diseased_Orange (:,:,1);
R_Pest_Orange = Pest_Orange (:,:,1);
G_Healthy_Orange = Healthy_Orange (:,:,2);
G_Diseased_Orange = Diseased_Orange (:,:,2);
G_Pest_Orange = Pest_Orange (:,:,2);
B_Healthy_Orange = Healthy_Orange (:,:,3);
B_Diseased_Orange = Diseased_Orange (:,:,3);
B_Pest_Orange = Pest_Orange (:,:,3);
```

A4. MATLAB Code Contrast Adjustment of Images Grayscale Images Shown in Fig. 4.7

```
Adj_Healthy_Orange_GreyScale1 = imadjust(Healthy_Orange_GreyScale);
```

```
Adj_Diseased_Orange_GreyScale1 = imadjust(Diseased_Orange_GreyScale);
Adj_Pest_Orange_GreyScale1 = imadjust(Pest_Orange_Greyscale);
Adj_Healthy_Orange_GreyScale2 = histeq(Healthy_Orange_GreyScale);
Adj_Diseased_Orange_GreyScale2 = histeq(Diseased_Orange_GreyScale);
Adj_Pest_Orange_GreyScale2 = histeq(Pest_Orange_Greyscale);
```

A5. MATLAB for Otsu's Segmentation Method on the Grayscale images shown in Fig. 4.7

```
Otsu_Tresh_Healthy_Orange = graythresh(Adj_Healthy_Orange_GreyScale);
Otsu_Tresh_Diseased_Orange = graythresh(Adj_Diseased_Orange_GreyScale);
Otsu_Tresh_Pest_Orange = graythresh(Adj_Pest_Orange_GreyScale);
Binary_Healthy_Orange = imbinarize(Adj_Healthy_Orange_GreyScale...
,Otsu_Tresh_Healthy_Orange);
Binary_Diseased_Orange =imbinarize(Adj_Diseased_Orange_GreScale...
,Otsu_Tresh_Diseased_Orange);
Binary_Pest_Orange = imbinarize(Adj_Pest_Orange_GreyScale,Otsu_Tresh_Pest_Orange);
```

A6. MATLAB Code for Training an Orange Classifier

```
OrangeDataSet =
imageDatastore("dataset","IncludeSubfolders",true,"LabelSource","foldernames");
[TrainingData, ValidationData] = splitEachLabel(OrangeDataSet, 7.0);
ClassificationNetwork = googlenet;
DataInputSize = ClassificationNetwork.Layers(1).InputSize(1:2);
ResizedTrainingData = augmentedImageDatastore(DataInputSize,TrainingData);
ResizedValidationData = augmentedImageDatastore(DataInputSize,ValidationData);
FeatureLearner = ClassificationNetwork.Layers(142);
OutputClassifier = ClassificationNetwork.Layers(144);
NumberClasses = numel(categories(TrainingData.Labels));
OrangeFeatureLearner = fullyConnectedLayer(NumberClasses,"Name","Orange Feature
Learner",...
   "WeightLearnRateFactor",10, "BiasLearnRateFactor",10);
OrangeClassifierLayer = classificationLayer ("Name", "Orange Classifier");
Layer_Graph = layerGraph(ClassificationNetwork);
```

```
New_Layer_Graph =
replaceLayer(Layer_Graph,FeatureLearner.Name,OrangeFeatureLearner);
New_Layer_Graph =
replaceLayer(New_Layer_Graph,OutputClassifier.Name,OrangeClassifierLayer);
Minibatch =10;
ValidationFrequency = floor(numel(ResizedValidationData.Files)/Minibatch);
Training_Option = trainingOptions("sgdm","MiniBatchSize",Minibatch,"MaxEpochs",10 ...
   , "InitialLearnRate",3e-4, "Shuffle", 'every-epoch',
"ValidationData",ResizedValidationData,...
   "ValidationFrequency",ValidationFrequency, "Verbose", false, "Plots", "training-
progress");
ClassificationNetwork = train
network(ResizedTrainingData,New_Layer_Graph,Training_Option);
```

A7. MATLAB Code for Testing an Orange Classifier

```
the_camera = webcam;
RealTimeImage = figure;
while ishandle (RealTimeImage)
      single_image = snapshot(the_camera);
      Resized_the_camera = imresize(single_image,DataInputSize);
      image(Resized_the_camera)
      [predicted_item, probability] = classify(ClassificationNetwork, Resized_the_camera);
      title({char(predicted_item),num2str(max(probability),2)});
      drawnow;
end
```

A8. MATLAB Code for Acquiring the Input Images and Testing Proposal 2

```
the_camera = webcam(1);
RealTimeImage = figure;
while ishandle (RealTimeImage)
      single_image1 = snapshot(the_camera);
      single_image2 = snapshot(the_camera);
```

```
    Resized_single_image1 = imresize(single_image1,DataInputSize);
    Resized_single_image2 = imresize(single_image2,DataInputSize);
    [predicted_item1, probability1] = classify(ClassificationNetwork,
Resized_single_image1);
    [predicted_item2, probability2] = classify(ClassificationNetwork,
Resized_single_image2);
    subplot(1,2,1),imshow(Resized_single_image1);
    title({char(predicted_item1),num2str(max(probability1),2)})
    subplot(1,2,2),imshow(Resized_single_image2);
     title({char(predicted_item2),num2str(max(probability2),2)});
    drawnow;
    if isequal( predicted_item1, 'black-rot affected oranges')
       fprintf('Diseased Orange Detected!\n');
       fprintf('Final Classification = Black rot affected orange!\n');
    elseif isequal( predicted_item2, 'black-rot affected oranges')
       fprintf('Diseased Orange Detected!\n');
       fprintf('Final Classification = Black rot affected orange!\n');
    else
      fprintf('Healthy Orange Detected!\n');
       fprintf('Final Classification = Healthy orange!\n');
    end
end
```

A9. MATLAB Code for Acquiring Input Images and Testing Proposal 3

```
the_camera1 = webcam(1);
the_camera2 = webcam(2);
RealTimeImage = figure;
while ishandle (RealTimeImage)
    single_image1 = snapshot(the_camera1);
```

```
    single_image2 = snapshot(the_camera2);
    Resized_single_image1 = imresize(single_image1,DataInputSize);
    Resized_single_image2 = imresize(single_image2,DataInputSize);
    [predicted_item1, probability1] = classify(ClassificationNetwork,
Resized_single_image1);
    [predicted_item2, probability2] = classify(ClassificationNetwork,
Resized_single_image2);
    subplot(1,2,1),imshow(Resized_single_image1);
    title({char(predicted_item1),num2str(max(probability1),2)})
    subplot(1,2,2),imshow(Resized_single_image2);
     title({char(predicted_item2),num2str(max(probability2),2)});
    drawnow;
    if isequal( predicted_item1, 'black-rot affected oranges')
       fprintf('Diseased Orange Detected!\n');
       fprintf('Final Classification = Black rot affected orange!\n');
    elseif isequal( predicted_item2, 'black-rot affected oranges')
       fprintf('Diseased Orange Detected!\n');
       fprintf('Final Classification = Black rot affected orange!\n');
    else
      fprintf('Healthy Orange Detected!\n');
       fprintf('Final Classification = Healthy orange!\n');
    end
end
```

A10. MATLAB Code for Training and Validation of Proposal 4

```
FruitDataSet =
imageDatastore("DATASET2","IncludeSubfolders",true,"LabelSource","foldernames");
[TrainingData, ValidationData] = splitEachLabel(FruitDataSet, 7.0);
ClassificationNetwork = googlenet;
DataInputSize = ClassificationNetwork.Layers(1).InputSize(1:2);
ResizedTrainingData = augmentedImageDatastore(DataInputSize,TrainingData);
ResizedValidationData = augmentedImageDatastore(DataInputSize,ValidationData);
FeatureLearner = ClassificationNetwork.Layers(142);
```

```
OutputClassifier = ClassificationNetwork.Layers(144);
NumberClasses = numel(categories (TrainingData.Labels));
NewFeatureLearner = fullyConnectedLayer(NumberClasses,"Name","Fruit Feature
Learner","WeightLearnRateFactor",10,"BiasLearnRateFactor",10);
NewClassifierLayer = classificationLayer ("Name","Hybrid-DQMS Classifier");
Layer_Graph = layerGraph(ClassificationNetwork);
New_Layer_Graph =
replaceLayer(Layer_Graph,FeatureLearner.Name,NewFeatureLearner);
New_Layer_Graph = replaceLayer
(New_Layer_Graph,OutputClassifier.Name,NewClassifierLayer);
Minibatch = 10;
ValidationFrequency = floor (numel(ResizedValidationData.Files)/Minibatch);
Training_Option = trainingOptions("sgdm","MiniBatchSize",Minibatch,"MaxEpochs",10 ...
   , "InitialLearnRate",3e-4,"Shuffle",'every-
epoch',"ValidationData",ResizedValidationData,"ValidationFrequency",
ValidationFrequency, "Verbose",false,"Plots", "training-progress");
ClassificationNetwork = trainNetwork
(ResizedTrainingData,New_Layer_Graph,Training_Option);
```

A.11. MATLAB Code for Testing Proposal 4

```
the_camera = webcam;
RealTimeImage = figure;
while ishandle (RealTimeImage)
      single_image = snapshot(the_camera);
      Resized_the_camera = imresize(single_image,DataInputSize);
      image(Resized_the_camera)
      [predicted_item, probability] = classify(ClassificationNetwork1, Resized_the_camera);
      title({char(predicted_item),num2str(max(probability),2)});
      drawnow;
  Signal_values = {0; 0; 0};
   actuator_signal = char(predicted_item);
switch actuator_signal
   case 'Healthy orange'
```

```
        Signal_values = {0; 0; 0};
        disp('Healthy Orange Detected');
        disp(Signal_values);
    case 'Healthy apple'
        Signal_values = {0; 0; 1};
        disp('Healthy Apple Detected');
        disp(Signal_values);
    case {'Blotch affected apple','Black-rot affected orange'}
        Signal_values = {0; 1; 0};
        fprintf('Diseased Fruit Detected!\n');
        fprintf('Actuator1_Actuator2_Actuator3_signals =\n');
        disp(Signal_values);
    case {'Damaged apple', 'Damaged orange'}
        Signal_values = {1; 0; 0};
        disp('Damaged Fruit Detected');
        disp(Signal_values);
end
end
```