

A MULTISPECTRAL AND MACHINE LEARNING
APPROACH TO EARLY STRESS CLASSIFICATION
IN PLANTS

Submitted in fulfilment
of the requirements of the degree of

MASTER OF SCIENCE

of Rhodes University

Louise Poole

Grahamstown, South Africa

March 12, 2022

Declaration of Authorship

I, Louise Poole, declare that this thesis titled, *A Multispectral and Machine Learning Approach to Early Stress Classification in Plants* and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or other qualification at this University or any other institution, this is clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have clarified what others did and what I have contributed to myself.

Signed:

Date:

Abstract

Crop loss and failure can impact both a country's economy and food security, often to devastating effects. As such, the importance of successfully detecting plant stresses early in their development is essential to minimize spread and damage to crop production. Identification of the stress and the stress-causing agent is the most critical and challenging step in plant and crop protection. With the development of and increase in ease of access to new equipment and technology in recent years, the use of spectroscopy in the early detection of plant diseases has become notably popular.

This thesis narrows down the most suitable multispectral imaging techniques and machine learning algorithms for early stress detection. Datasets were collected of visible images and multispectral images. Dehydration was selected as the plant stress type for the main experiments, and data was collected from six plant species typically used in agriculture. Key contributions of this thesis include multispectral and visible datasets showing plant dehydration as well as a separate preliminary dataset on plant disease.

Promising results on dehydration showed statistically significant accuracy improvements in the multispectral imaging compared to visible imaging for early stress detection, with multispectral input obtaining a 92.50% accuracy over visible input's 77.50% on general plant species. The system was effective at stress detection on known plant species, with multispectral imaging introducing greater improvement to early stress detection than advanced stress detection. Furthermore, strong species discrimination was achieved when exclusively testing either early or advanced dehydration against healthy species.

ACM Computing Classification System

Thesis classification under the ACM Computing Classification System¹ (2012 version valid through 2022):

- **Computing methodologies ~ Neural networks**
- **Computing methodologies ~ Object identification**
- *Applied computing ~ Agriculture*

General-Terms: Convolutional Neural Networks, Image Classification, Multispectral Imaging, Plant Stress Detection.

¹<https://www.acm.org/publications/class-2012>

Acknowledgements

The authors would like to thank the authors of the publicly available datasets used in this paper. This work was undertaken in the Distributed Multimedia CoE at Rhodes University, with financial support from Telkom SA and CORIAN. The authors acknowledge that opinions, findings and conclusions or recommendations expressed here are those of the author(s) and that none of the above mentioned sponsors accept liability whatsoever in this regard.

I would also like to acknowledge all those who directly and indirectly supported me through my work. In particular I would like to thank my supervisor, Dr. Dane Brown, for his support, encouragement and advice throughout the project as well as James Connan and his invaluable input. Finally, I would like to thank my friends and family for their emotional support and encouragement throughout this project.

Contents

1	Introduction	1
1.1	Context of Research	1
1.2	Motivation for this Research	2
1.3	Problem Statement	2
1.4	Research Question	3
1.5	Research Objectives	3
1.6	Approach	4
1.7	Assumptions	4
1.8	Limitations	5
1.9	Thesis Outline	5
2	Concepts and Literature Review	7
2.1	Plant Stress and Disease Classification	7
2.2	Multispectral Imaging	8
2.2.1	Spectral Reflectance of Plants	9
2.2.2	Related Studies	10
2.2.2.1	Wavelength Studies	10

2.2.2.2	Dataset Collection	12
2.3	Machine Learning Approaches	14
2.3.1	CNNs	17
2.3.1.1	ResNet	19
2.3.1.2	VGG	20
2.3.1.3	GoogLeNet	21
2.3.2	Improving Model Generalization	22
2.3.2.1	Data Augmentation	23
2.3.2.2	Dropout	23
2.3.2.3	Batch Normalization	24
2.3.2.4	Early Stopping	25
2.3.3	Parameter Tuning	25
2.3.3.1	Grid search	26
2.3.3.2	Random Search	26
2.3.3.3	Hyperband	28
2.3.3.4	Bayesian Optimization	28
2.4	Summary	30
3	Methodology	32
3.1	Methodology Overview	32
3.1.1	CNN Architecture Selection	32

3.1.2	Plant Stress Classification System	33
3.2	Dataset Construction	34
3.2.1	Mode of VIS-IR Spectroscopy	35
3.2.2	Equipment	36
3.2.3	Image Composition	37
3.2.4	Acquisition of Moisture Readings	38
3.2.5	Division of Classes	38
3.3	Summary	39
4	Implementation	40
4.1	Data Collection	40
4.1.1	Plant Species	40
4.1.2	Equipment	41
4.1.3	Allocation of Classes	41
4.2	Machine Learning Models	42
4.2.1	System Infrastructure	42
4.2.1.1	Hardware	43
4.2.1.2	Software	43
4.2.2	Architectures	43
4.2.2.1	ResNet-44	43
4.2.2.2	VGG-16	45

4.2.2.3	GoogLeNet	46
4.2.3	Improvement of Model Generalization	48
4.2.3.1	Data Augmentation	48
4.2.3.2	Early Stopping	49
4.2.4	Parameter Tuning	49
4.2.5	Training and Evaluation	50
4.3	Summary	51
5	Experimental Setup	53
5.1	Measuring Model Performance	53
5.1.1	Accuracy	53
5.1.2	F-score	54
5.1.3	McNemar Test	55
5.2	Test Models	57
5.2.1	CNN Architecture Evaluation	57
5.2.2	Early Stress Detection	57
5.2.2.1	Individual Plant Species	58
5.2.2.2	All Plant Species	58
5.3	Datasets	59
5.3.1	Publicly Available Datasets	59
5.3.1.1	Mendeley Leaves Dataset	59

5.3.1.2	PlantVillage Dataset	61
5.3.2	Collected Datasets	64
5.3.2.1	Disease Dataset	64
5.3.2.2	Early and Advanced Stress Dataset	65
5.4	Summary	68
6	Results and Discussion	69
6.1	Evaluation of CNN Architecture	69
6.1.1	Parameter Tuning	69
6.1.2	Experiment 1.1	71
6.1.3	Experiment 1.2	73
6.1.4	Experiment 1.3	74
6.1.5	Summary of Experiment 1	75
6.2	Early Water Stress Detection	75
6.2.1	Individual Plant Species	75
6.2.1.1	Experiment 2.1	76
6.2.1.2	Experiment 2.2	77
6.2.1.3	Experiment 2.3	78
6.2.1.4	Summary of Experiment 2	80
6.2.2	All Plant Species	80
6.2.2.1	Experiment 3.1	80

6.2.2.2	Experiment 3.2	81
6.2.2.3	Experiment 3.3	82
6.2.2.4	Summary of Experiment 3	83
6.2.2.5	Experiment 4.1	83
6.2.2.6	Experiment 4.2	84
6.2.2.7	Experiment 4.3	85
6.2.2.8	Summary of Experiment 4	87
6.3	Summary	87
7	Conclusion and Future Work	89
7.1	Conclusion	89
7.2	Contributions	91
7.3	Future Work	92
	Appendices	103
A	Additional Results	103
A.1	Experiment 2.1	103
A.2	Experiment 2.2	104
A.3	Experiment 2.3	105
B	Datasets	106
B.1	Spinach Disease Dataset	106
B.2	Early and Advanced Dehydration Dataset	107

List of Figures

2.1	Electromagnetic spectrum [CC BY-SA 3.0 - edited].	9
2.2	The hierarchy of AI.	15
2.3	A single neuron within a neural network.	16
2.4	Deep learning network.	17
2.5	The ResNet residual block.	19
2.6	The VGG architectures for 11, 13, 16 and 19 layers.	21
2.7	The GoogLeNet inception block with the split, transform and merge concept.	22
2.8	The application of 50% dropout to a three-layer network.	24
2.9	Graphical representation of the search area in a grid search [CC BY-SA 4.0].	27
2.10	Graphical representation of the search area in a random search [CC BY-SA 4.0].	28
2.11	Graphical representation of the search area in Bayesian Optimization [CC BY-SA 4.0].	30
3.1	High-level view of the early stress detection system.	34
3.2	Modes of VIS-IR spectroscopy.	36
4.1	Composition of the ResNet-44 network implemented.	44

5.1	Sample images of each plant species in the collected dataset.	67
6.1	Training and validation accuracy of the ResNet-44 model trained on Mende- ley Leaves dataset.	70
6.2	Training and validation accuracy of the GoogLeNet model trained on Mende- ley Leaves dataset.	71
6.3	Confusion matrix of the ResNet-44 model trained and tested on the Mende- ley Leaves dataset. Lighter colors indicate higher concentrations of test predictions with correct predictions found along the diagonal.	72
6.4	Accuracies of the multispectral and visible systems classifying <i>healthy, early dehydration</i> and <i>advanced dehydration</i> on separate plant species.	76
6.5	Confusion matrix produced by the multispectral spinach model in Experi- ment 2.1. Lighter colors indicate higher concentrations of test predictions with correct predictions found along the diagonal.	77
6.6	Accuracies of multispectral and visible systems classifying healthy and ad- vanced dehydration on separate plant species.	78
6.7	Accuracies of multispectral and visible systems classifying healthy and early dehydration on separate plant species.	79
6.8	Confusion matrix produced by the multispectral model in Experiment 4.3. Lighter colors indicate higher concentrations of test predictions with correct predictions found along the diagonal.	86
B.1	Samples from each class of the spinach diseased visible and infrared col- lected datasets.	106
B.2	Samples from each class of the cauliflower visible and infrared collected datasets.	107

-
- B.3 Samples from each class of the jalapeno visible and infrared collected datasets. 107
- B.4 Samples from each class of the lettuce visible and infrared collected datasets. 108
- B.5 Samples from each class of the rocket visible and infrared collected datasets. 108
- B.6 Samples from each class of the spinach visible and infrared collected datasets. 109
- B.7 Samples from each class of the tomato visible and infrared collected datasets. 109

List of Tables

3.1	Required equipment and its intended uses.	37
4.1	Number of plants per plant species in the collected dataset.	41
4.2	Soil and leaf moisture content readings used to separate the data into target classes.	42
4.3	Filter sizes and number of repeats of the residual block in ResNet-44. . . .	44
4.4	Filter sizes of the convolutional blocks in VGG-16.	46
5.1	Example contingency table for McNemar’s test.	56
5.2	Breakdown of the Mendeley Leaves image dataset.	59
5.3	Breakdown of the PlantVillage image dataset.	61
5.4	Breakdown of the collected spinach disease image dataset.	64
5.5	Breakdown of the collected early and advanced dehydration image dataset. .	65
6.1	Best parameters determined with KerasTuner for evaluation of ResNet-44, GoogLeNet, and VGG-16 architectures.	70
6.2	Performance of architectures on balanced Mendeley Leaves dataset.	72
6.3	Performance of the architectures on the balanced PlantVillage dataset. . .	73
6.4	Performance of architectures on the collected spinach disease dataset. . . .	74

6.5	Performance of the multispectral and visible systems classifying healthy, early, and advanced dehydration on all plant species.	81
6.6	Performance of the multispectral and visible systems classifying healthy and advanced dehydration on all plant species.	81
6.7	Performance of the multispectral and visible systems classifying healthy and early dehydration on all plant species.	82
6.8	Distribution of test output of the multispectral and visible systems classifying early dehydration.	82
6.9	Performance of the multispectral and visible systems classifying plant species and healthy, early or advanced dehydration.	84
6.10	Performance of the multispectral and visible systems classifying plant species and healthy or advanced dehydration.	84
6.11	Performance of the multispectral and visible systems classifying plant species and healthy or early dehydration.	85
A.1	Performance of the models classifying healthy, early dehydration and advanced dehydration on separate plant species.	103
A.2	Performance of the binary models classifying healthy and advanced dehydration on separate plant species.	104
A.3	Performance of the binary models classifying healthy and early dehydration on separate plant species.	105

Listings

4.1	ResNet-44 residual block.	45
4.2	Block 5 of VGG-16.	46
4.3	GoogLeNet Inception block.	46
4.4	Data augmentation layers.	48
4.5	Data augmentation using ImageDataGenerator.	49
4.6	Early stopping callback.	49
4.7	Parameter tuning with KerasTuner.	50
4.8	Model training.	51
4.9	Model training.	51

Declaration of Publications

The following research papers have been accepted and presented for publication, and parts of its materials are included in the thesis:

1. **Louise Poole** and Dane Brown. Investigating popular CNN architectures for plant disease detection. In *2021 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, pages 1–5. IEEE, 2021.
2. **Louise Poole**, Dane Brown, James Connan. Early dehydration detection using infrared imaging. In *2021 Southern Africa Telecommunication Networks and Applications Conference (SATNAC)*. 2021.

Moreover, a journal titled: “Investigating Computer Vision Approaches to Multiclass Plant Disease Identification” was submitted in 2021 and is under review.

Glossary

AI	Artificial Intelligence
BN	Batch Normalization
CNN	Convolutional Neural Network
DL	Deep Learning
EM	Electromagnetic Spectrum
ICS	Internal Covariate Shift
IR	Infrared
k-NN	K-Nearest Neighbour
ML	Machine Learning
NIR	Near Infrared
NN	Neural Network
PNN	Probabilistic Neural Network
SVM	Support Vector Machine
VIS	Visible
VIS-IR	Visible and Infrared
VIS-NIR	Visible and Near Infrared

1

Introduction

1.1 Context of Research

Crop loss and failure can impact both a country's economy and food security, often to devastating effects (Dean *et al.*, 2005). For example, the rice blast disease alone annually destroys enough rice to feed the entire population of Italy. Crop loss can occur as a result of many factors including, but not limited to disease, drought and pests. This constant pressure placed on the agriculture sector only increases during plant epidemics.

Once discovered, damaged or infected plants should be removed, and in the case of many plants such as grapes and vineyards, those plants must be replaced with new ones. These new plants often take many years to reach their full production potential. Essentially, plant loss potentially affects many years of food production. As such, the importance of successfully detecting plant stresses early in their development is essential to minimize spread and damage to crop production (Yang *et al.*, 2013).

Early identification of plant stress is vital in almost all cases to prevent plant or crop loss and disease spread. Identification of the stress and the stress-causing agent is the most critical and challenging step in plant and crop protection. Automating the process using computer vision and machine learning techniques could provide an accurate and more accessible means for plant and crop stress identification.

1.2 Motivation for this Research

Identifying and rating diseases is an expensive, time consuming, subjective and unreliable practice (Bock *et al.*, 2010). It is traditionally done by hand. Automating the process using computer vision practices and machine learning techniques can potentially negate many disadvantages of conducting disease identification by hand. In computer vision, image segmentation plays a significant role in image content understanding and visual entity recognition (Janwale, 2017).

With the development of and increase in ease of access to new equipment and technology in recent years, the use of spectroscopy in the early detection of plant diseases has become notably popular. This is due to the increase in demand for inexpensive, rapid and non-invasive applications for plant stress detection and classification (López *et al.*, 2003, Zovko *et al.*, 2019).

While VIS-IR has been extensively studied and effectively applied to systems designed to determine the quality of agricultural products (Sun, 2009), few studies focus on its application in the detection of plant stress and diseases (Khaled *et al.*, 2018, Martinelli *et al.*, 2015). Furthermore, there is a lack of literature determining its effectiveness in detecting the early stages of stress and diseases in plants.

1.3 Problem Statement

This research investigates the applicability of combining multispectral imaging with machine learning techniques. Their effectiveness at detecting and classifying the stress state of plant leaves in images can be evaluated against visible imaging.

The hypothesis is that a system can be constructed to identify early plant stress symptoms with improved accuracy by leveraging multispectral imaging instead of visible imaging.

1.4 Research Question

Based on the above, the following overarching research question can be formulated: ‘How effective is multispectral imaging compared to visible imaging for early plant stress classification?’ This main research question can be broken into the following sub-questions:

1. Does multispectral input provide improved performance over visible input for early stress classification?
2. Is multispectral input effective on multiple stress types?
3. When does multispectral input enable the largest accuracy gains:
 - (a) with early stress classification?
 - (b) with advanced stress classification?
 - (c) with multitask classification of plant species too?
 - (d) with separate plant species?

1.5 Research Objectives

This research aims to achieve the following objectives:

1. Progressively collect visible and multispectral image datasets on stress in plant species in a real-world setting.
2. Design and construct a relatively inexpensive system that classifies plant species and their individual stage of stress.
3. Conduct an experiment that measures the performance of state-of-the-art CNN architectures when trained on a plant disease image dataset.
4. Conduct an experiment that measures the effectiveness of visible and multispectral input when classifying *healthy* versus *advanced* stress on multiple plants species.

5. Conduct an experiment that measures the effectiveness of visible and multispectral input when classifying *healthy* versus *early* stress on multiple plant species.
6. Conduct an experiment that measures the effectiveness of visible and multispectral input when classifying *healthy*, *early* and *advanced* stress on multiple plant species.
7. Conduct an experiment that measures the effectiveness of visible and multispectral input with multitask classification *plant species* as well as state of *healthy*, *early* and *advanced* stress.

1.6 Approach

The summary approach to this thesis is as follows. Various machine learning algorithms, multispectral wavelength ranges and data collection techniques are analyzed in the literature. The best theoretical approaches are identified based on their effectiveness on plant-based data and their viability for deployment in the field.

The success of the resulting plant stress detection and classification system is measured as follows. Machine learning model metrics appropriate to the dataset are chosen to evaluate the system. Classifier parameters are systematically tuned for optimal models per experiment. Experiments are conducted on the collected image dataset to compare the effectiveness of visible and far-red/near-infrared (multispectral) input per individual plant species and general plant species.

1.7 Assumptions

Data collection for the experiments is conducted under the following assumptions:

- all plants are healthy and hydrated for the start of data collection.

- the predominant stress experienced by the plants during the data collection series is early or advanced dehydration, as this is the main experiment. However, disease stress is also evaluated in preliminary experiments.

1.8 Limitations

- Quality standards for images containing plants do not exist, and thus, image quality is subjectively evaluated during experimentation using visual inspection.
- Since the second objective is to create an inexpensive and easily applicable system compared to stress classification by hand, costly and complicated data acquisition procedures will be disregarded — inexpensive and easily accessible consumer computer hardware should suffice.

1.9 Thesis Outline

The remainder of this thesis is arranged as follows:

Chapter 2: *Concepts and Literature Review*: A broad overview of plant stress literature including disease classification, multispectral imaging and machine learning are given. The concepts, algorithms and data used in related studies are also discussed and explained.

Chapter 3: *Methodology*: This chapter presents and discusses the proposed approach to create an automated early stress classification system.

Chapter 4: *Implementation*: This chapter presents the implementation details for the proposed classification system and data collection procedures.

Chapter 5: *Experimental Setup*: This chapter details the experimental setup used in this research. It begins with the appropriate metric selection followed by the design of all experiments conducted, and finally, the breakdown of all the image datasets utilized in the experiments.

Chapters 6: *Results and Discussion*: The validation of the machine learning architecture used in the system implementation is presented, followed by the results achieved by the proposed system during testing and an analysis and discussion on the test results.

Chapter 7: *Conclusion*: This chapter concludes the thesis, highlights the contributions made towards the research and provides directions for future work.

2

Concepts and Literature Review

This chapter gives a broad overview of plant stress and disease classification and multispectral imaging. It details the concepts and techniques used in related studies and provides the required knowledge for automating the plant stress and disease classification process. Particular focus is placed on early stress and disease detection.

2.1 Plant Stress and Disease Classification

Due to climate change and rising population numbers, food production and food security have become increasingly important. One method to prevent crop loss is to develop a system that allows for early, efficient and accurate identification of plant stress and disease. However, plant stress and disease identification is challenging due to the varying nature of the visual appearance of different diseases and stress on diverse plant species. Many previously proposed methods utilize a combination of image processing with machine learning to achieve this.

Previously popular machine learning algorithms such as k-Nearest Neighbours (k-NN), Support Vector Machines (SVM), Naive Bayes and Probabilistic Neural Networks (PNN) perform well on plant disease data (Harish *et al.*, 2013, Priya *et al.*, 2012, Singh *et al.*, 2010), but are significantly outperformed by Convolutional Neural Networks (CNNs) (Lu *et al.*, 2021, Mohanty *et al.*, 2016). It is also noted that unlike the other machine learning

algorithms listed, CNNs perform well without requiring accurate feature extraction prior to model training (Rawat and Wang, 2017).

Many CNN architectures have been developed to perform well on image datasets. The performance of each architecture depends on the nature of the dataset under evaluation. The strong classification performance of a particular architecture on a given dataset does not necessarily imply the excellent performance of the same architecture on a dataset of a different nature.

Furthermore, with the development of more accessible and newer equipment and technology, the use of spectroscopy in the early detection of plant stress and disease has become notably popular. This is due to the increase in demand for inexpensive, rapid and non-invasive applications for plant stress and disease detection and classification (Khaled *et al.*, 2018, López *et al.*, 2003). Early disease and stress detection allow for timely intervention before critical thresholds are exceeded and plant productivity is affected.

Visible and Infrared (VIS-IR) spectroscopy has been extensively studied and effectively applied to systems designed to determine the quality of agricultural products (Sun, 2009). However, few studies focus on its use in plant stress and disease detection (Khaled *et al.*, 2018, Martinelli *et al.*, 2015). Furthermore, there is a lack of literature for determining VIS-IR's effectiveness in *early* disease and stress detection.

2.2 Multispectral Imaging

Spectroscopy is the study of electromagnetic (EM) waves and their interaction with matter. Electromagnetic waves include the ultraviolet (UV), visible (VIS) and infrared (IR) spectra (Pavia *et al.*, 2014), as shown in Figure 2.1 (Verhoeven, 2017). Multispectral imaging is an analytical technique based on spectroscopy where images are captured within specific wavelength ranges across the electromagnetic spectrum. The majority of multispectral sensors used in agriculture for remote sensing capture visible (VIS) and near-infrared (NIR) regions of the EM spectrum (Navulur, 2006).

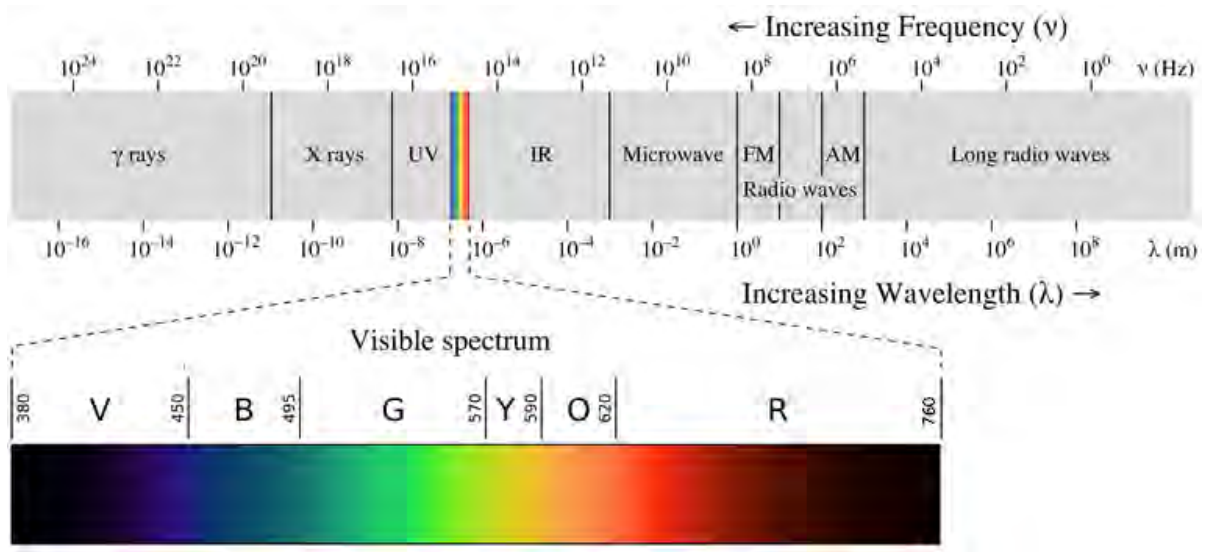


Figure 2.1: Electromagnetic spectrum [CC BY-SA 3.0 - edited].

Visible and infrared (VIS-IR) spectroscopy, with wavelengths 380-1000000 nm (Pitts and Kleinstein, 1993) is considered one of the most promising and extensively accepted non-invasive techniques used for plant analysis (Khaled *et al.*, 2018). Its widespread acceptance stems from the many advantages over other analytical techniques. Current literature mainly focuses on the use of VIS-IR imaging for determining the quality of agricultural produce (Su and Sun, 2018) and does not effectively explore the combination of computer vision and VIS-IR imaging for its use in the detection of plant stress and diseases (Khaled *et al.*, 2018, Martinelli *et al.*, 2015).

2.2.1 Spectral Reflectance of Plants

Research has shown a relationship between specific absorbance-reflectance patterns for the photoactive pigments in a leaf, mainly chlorophylls and carotenoids, and the physiological state of the leaf (Sims and Gamon, 2002).

Both of these photoactive pigments play a vital role in a plant's ability to photosynthesize. Chlorophylls and carotenoids collaborate to absorb and convert the light energy that reaches the leaf (Amthor, 2010). Changes in chlorophyll concentration and proportions are triggered by stress, ageing and other factors directly related to the production rate of

the plant (Blackburn and Ferwerda, 2008). Therefore, significant changes in reflectance in the leaf due to stress caused by strain or pathogens and diseases can be observed. The visible range 380-760 nm and the infrared range 760-1 000 000 nm (Pitts and Kleinstein, 1993) are sensitive to these changes to different degrees. Furthermore, the far-red 690-760 nm and NIR 760-1400 nm ranges (Pitts and Kleinstein, 1993) are typically more sensitive to early infection and indicate observable changes before other regions of the electromagnetic spectrum do (Carter and Knapp, 2001, Gitelson *et al.*, 2003, Jones and Vaughan, 2010).

Specific wavelength ranges that are sensitive to the stress experienced by plants depend on particular combinations of stress and plant species in question (Sims and Gamon, 2002). Stresses and diseases found within various plant species are thus explored in Section 2.2.2.1.

2.2.2 Related Studies

The following studies relate to the implementation of spectroscopy in plant stress and disease identification.

2.2.2.1 Wavelength Studies

Many studies have been conducted to determine the wavelengths that experience the most significant changes in leaf reflectance when infected or stressed for various plant species. The following studies compare the VIS and NIR ranges and smaller ranges found within them.

Ausmus and Hilty (1971) studied both the maize dwarf mosaic virus and *Helminthosporium maydis* diseases and their effect on the reflectance of corn leaves. They found that infected and healthy leaves possessed similar reflectance in the visible range of 400-750 nm but significantly different reflectance in the NIR range of 800-2600 nm.

Malthus and Madeira (1993) studied the reflectance properties of field bean infected by the fungus *Botrytis fabae* over the VIS-IR wavelength range of 400-1100 nm. They found that wavelengths around 800 nm produced the most significant response to the changes caused within the leaf by the fungus.

Peñuelas *et al.* (1993) studied the use of reflectance bands as an indicator of the water status of gerbera, pepper and bean plants. They observed that significant changes in the 950-970 nm narrow wavelength range were evident when the water stress¹ of the plants became well-developed. They noted that the maximum response within that wavelength range was found in the plant species that lost cell wall elasticity in response to drought stress. However, relative water content was identified as an essential factor, as shown by the highest correlation coefficients with spectral indices obtained from the 950-970 nm wavelength range.

Riedell and Blackmer (1999) studied the effect of Russian wheat aphids and greenbugs on the reflectance spectra of wheat. They determined that the narrow band wavelengths most responsive to the crop stress caused by these pests were 625-635 nm and 680-695 nm.

Bravo *et al.* (2003) studied yellow rust on the leaf canopy of wheat plants. They used four narrow wavebands from the VIS/NIR ranges of the spectrum, namely 543 ± 10 , 630 ± 10 , 750 ± 10 and 861 ± 10 , to create spectral profiles and calculate spectral indices, which they fed to a classification model. They obtained a 96% classification success rate between the healthy and infected wheat plants.

Xu *et al.* (2007) studied leaf miner damage on tomato leaf and found the short wavelengths of the NIR range of 800-1100 nm provided the most useful data for infection severity. When further narrowing down the bands, they found the individual bands of 1450 nm and 1900 nm to be the most applicable.

Zygielbaum *et al.* (2009) studied the reflectance properties of water stress² on greenhouse-grown maize. They found a strong correlation between the relative water content of the

¹drought stress was introduced through water deprivation.

²drought stress was introduced through water deprivation.

plant and the reflectance of the visible 400-700 nm wavelength range. They conclude that this wavelength range provides a means to estimate relative water content and detect early stages of plant stress.

Kim *et al.* (2010) studied the effect of plant stress on the reflectance of Buckeye Gala apple leaves. They used five different levels of water treatment on five groups of young apple trees in a greenhouse. Spectral images over a 400-1000 nm wavelength range were captured. They found that the range of 680-800 nm provided the strongest correlation to plant stress levels.

One limitation of the above-listed studies is that they use specialized and expensive equipment to gather their data. The equipment list of the studies above includes mounted spectrographs, specialized spectrometers etc., which are used to produce spectral profiles and indices of plant leaves. As specified in the research limitations in Section 1.8, this research focuses on broader wavelengths obtainable by more affordable visible and infrared cameras and relies on machine learning to make the final classification. Due to the majority of the aforementioned studies selecting various wavelengths in the far-red and NIR ranges, VIS-NIR input covering both ranges will be used in this research and will henceforth be referred to as ‘multispectral input’.

2.2.2.2 Dataset Collection

Nouri *et al.* (2018) created a dataset of close range near-infrared (NIR) hyperspectral images documenting the development of Apple Scab on apple leaves. Two series of images were taken, one of healthy leaves and one of infected leaves. Images were taken of the plants from within the controlled environment of a laboratory. A plant was inoculated with the disease, and images were taken daily from 2-11 days after inoculation. No pictures were taken beyond day 11 as the leaves were considered to have sufficiently developed advanced visual Apple Scab symptoms.

Limitations of their dataset include the following: only a single image was taken of a single leaf to represent each day in the build-up to infection, no data were collected on

days seven and eight, and only one disease on one plant species was considered. The methodology resulted in a tiny dataset of 16 images in total. Their dataset contains the desired input type and basic structure for the proposed research, but these limitations make the dataset impractical for deep learning applications.

A new dataset is required to be collected to address these limitations. Therefore, other plant image datasets are analyzed below to ascertain the procedures used in the existing literature.

[Hughes and Salathé \(2015\)](#) collected a plant disease image dataset known as PlantVillage. It is widely used for plant disease classification research.

The majority of the leaf images taken for this dataset were done by removing the leaf from the plant, placing it on a grey or black sheet of paper and taking an image. All images were taken outside with the sun as the light source. A range of light conditions encapsulated in the dataset replicated the range of lighting conditions likely to be experienced by the end-user. They used a standard point-and-shoot camera on automatic mode. They typically took 4-7 images per leaf while rotating the leaf 360 degrees. Some leaves were too large to capture in a single frame while retaining the small details, thus requiring multiple images of different sections of the leaf instead of rotating it. After initial collection, all images were processed by cropping away most of the background and orienting the leaves so that the tip of the leaf pointed upwards.

A total of 54 309 images were taken over 14 crop species and included 17 fungal diseases, 4 bacterial diseases, 2 mold diseases, 2 viral diseases and 1 disease caused by mites. Expert plant pathologists confirmed the identity of the diseases. More detail of the breakdown of this dataset is given in Section [5.3.1](#). One limitation of this dataset is that it does not cater for complex backgrounds. If the dataset is used to train a machine learning model, it would require future end-users to remove leaves from the plants and place them on a grey/black paper to allow their input to match the expected model input.

Upon a thorough investigation of the literature at the time of writing, the two datasets analyzed in this subsection contain relevant aspects for the proposed research. Combining

their data collection methodologies would produce the desired dataset for this research. Furthermore, [Nouri *et al.* \(2018\)](#)'s approach of collecting a series of multispectral images would enable the proposed dataset to be utilized for training machine learning models for early stress detection. [Hughes and Salathé \(2015\)](#)'s data collection methods provide the techniques for introducing the variation and large data quantities suited for training deep learning models. Capturing images with complex backgrounds while automating the image processing would prepare the final trained model for application in real-world scenarios.

2.3 Machine Learning Approaches

Machine learning (ML) is a subset of the artificial intelligence (AI) field and can be defined as the automated learning done by a program to make predictions without explicitly being programmed to make those predictions. Learning algorithms make this possible by using available data to estimate an unknown mapping between system inputs and outputs. Future predictions can thus be generated based on the previously observed samples. One of the most widely researched fields of machine learning is supervised learning. Data samples with both known inputs and known outputs are used during supervised learning ([Cunningham *et al.*, 2008](#)).

Furthermore, the models developed using supervised learning can generally be divided into three subcategories according to the task they are designed to accomplish: classification, regression and probability models. Classification models generate an estimation of class decision boundaries and are utilized when a limited number of distinct outputs are possible. Regression models generate real-valued outputs, and probability models generate probability density estimations ([Cherkassky and Mulier, 2007](#)).

There are many subfields in machine learning. Popular algorithms include neural networks (NN), decision trees, support vector machines, etc. A visual representation of this is shown in [Figure 2.2](#). The NN is inspired by the structure of the human brain and function and is referred to as neurocomputing. The brain is considered to be particularly good at tasks

such as pattern recognition, perception, flexible inference and intuition, all of which are valuable tasks to have executed by a computer (Anderson, 1995).

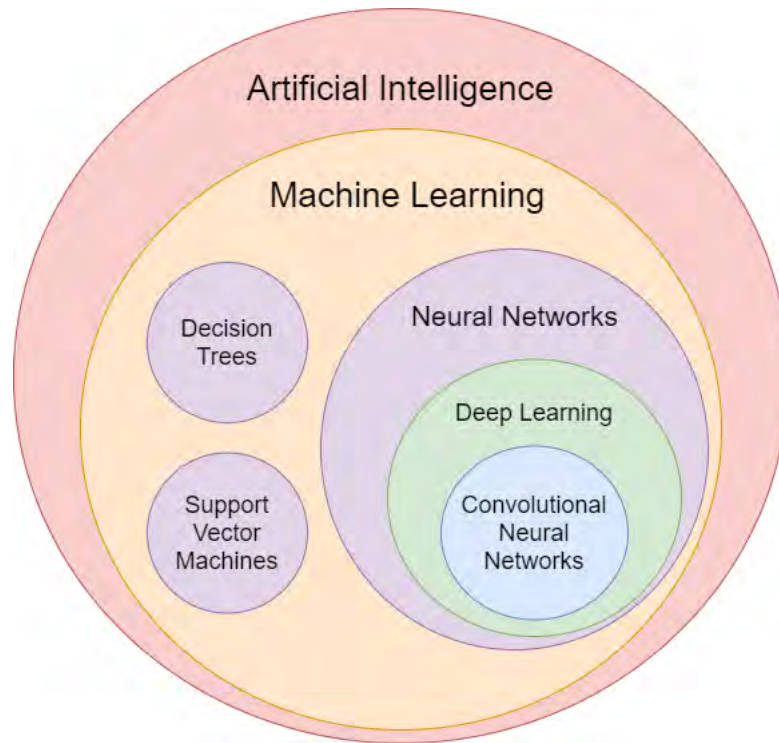


Figure 2.2: The hierarchy of AI.

Neural networks create an interconnected map of neurons. A visual representation of a neuron is given in Figure 2.3, where W is a matrix of the weights assigned to the connections attached to the neuron, x is a vector of the inputs sent to the neuron, and f and b are the activation function and bias of the neuron, respectively (Nielsen, 2015). The result of this equation is called the activation a of the neuron.

During training, both the weights W and bias b are updated. This enables the neuron to learn how to manipulate its input to produce the desired activation. The weights W are used to emphasize or depreciate the influence of the individual inputs received. However, the bias b is not influenced by previous layer input and alters the input of the activation function independently to the input x . Neurons are typically arranged in layers, with the output of one layer used as the input of the next layer. In a standard NN, each neuron in the previous layer is connected to every neuron in the next layer. This configuration is called a fully connected layer.

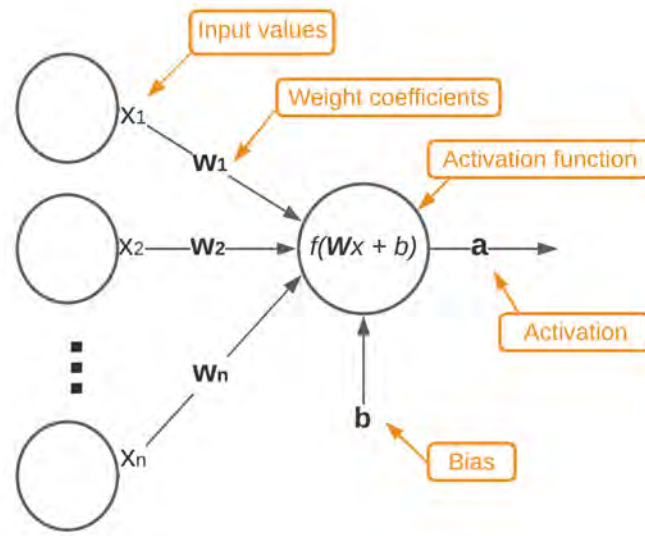


Figure 2.3: A single neuron within a neural network.

Traditional ML approaches require the application of feature extraction techniques to the data prior to the model receiving it. They focus the model, allowing robust feature learning by filtering confusing, unnecessary, unrelated information. The development of deep learning (DL) has significantly decreased the need for a model to rely on accurate feature extraction. It is a subfield of machine learning that is distinguishable by its use of at least three or more layers: an input layer, one or more hidden layers and an output layer, as demonstrated in Figure 2.4³. It is typically used on very large datasets. An advantage of DL is that relevant features are typically learned automatically, allowing explicit feature extraction steps to be omitted. In the fields of computer vision, speech recognition, bioinformatics, medical imaging, cybersecurity and many more, DL has been shown to outperform previously state-of-the-art machine learning approaches (Alom *et al.*, 2019).

Convolutional neural networks (CNN) is a deep learning algorithm that is particularly well-suited to image classification problems (Hussain *et al.*, 2018) and is explored in detail in Section 2.3.1.

³<https://www.ibm.com/za-en/cloud/learn/neural-networks> – Accessed 9 August 2021

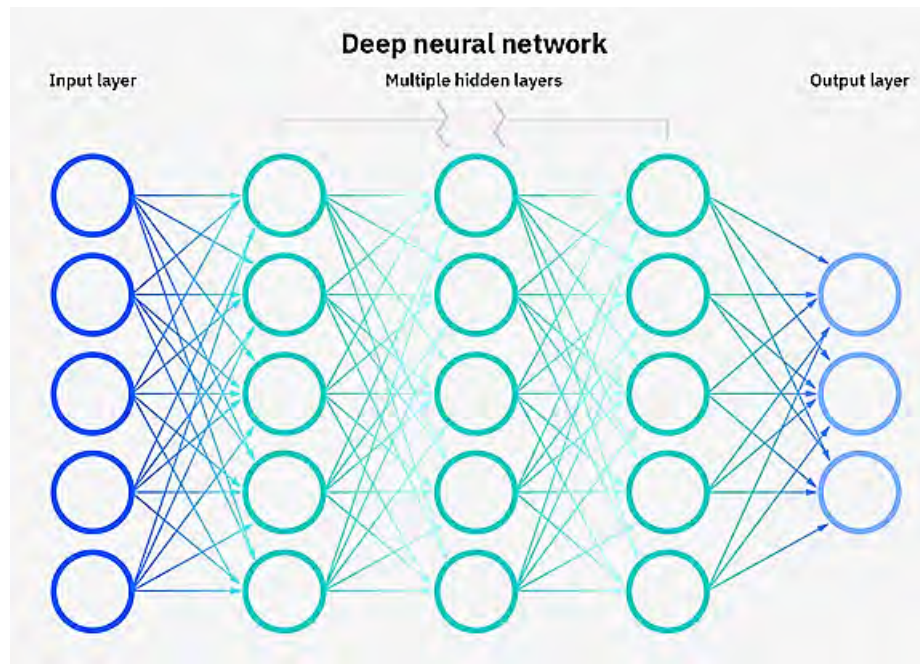


Figure 2.4: Deep learning network.

2.3.1 CNNs

A CNN is a deep neural network algorithm that utilizes convolutional layers. Convolutional layers are considered more specialized and efficient than fully connected layers.

In a fully connected layer, each neuron is connected to every previous layer's neuron. Each connection also has its own weight. This makes using fully connected layers computationally expensive as well as memory heavy.

On the other hand, in a convolutional layer, every neuron is connected to only a few local neurons in the previous layer. This proves to be very useful for spatial data where features are locally clustered ([Albawi et al., 2017](#)). It also effectively enables the CNN to prioritize salient features and ignore or prune unnecessary input. Therefore, feature extraction using image processing is often unnecessary for CNNs, whereas they are arguably compulsory for traditional classifiers to be effective on images with complex backgrounds.

In convolutional layers, the same set of weights is used for every neuron's set of connections. This is known as parameter sharing and is useful when features are equally likely to be found anywhere in the input — a common occurrence in images. Hence, CNNs are a

popular choice for learning objects in image datasets (Guo *et al.*, 2017). CNNs have also proven to be a typical choice for creating state-of-the-art plant stress and disease classification systems on a broad scope of diseases, stresses, and plant species (Loey *et al.*, 2020, Shruthi *et al.*, 2019).

A CNN architecture generally comprises a combination of one or more convolutional, pooling and fully connected layers. A few of the most popular architectures for image datasets include ResNet (He *et al.*, 2016), GoogLeNet (Szegedy *et al.*, 2015) and VGG (Simonyan and Zisserman, 2014). Examples of other well-known image classification CNN architectures that will not be explored in this research include AlexNet (Krizhevsky *et al.*, 2012), DenseNet (Huang *et al.*, 2017), and EfficientNet (Tan and Le, 2019) as they do not perform well on plant disease classification problems in the related literature. The performance of a CNN architecture is largely dependent on the dataset and type of data it is trained on (Domingos, 2012). Therefore, it is important to find and optimize an architecture that performs best on the specific problem or dataset in question.

An important practice when using CNNs is to obtain a training set that is as large as possible (Domingos, 2012). The many data points allow the algorithm to learn slight variations in the data and thus better generalize the problem. Smaller datasets can be expanded by using photometric and geometric transformations to augment the existing data. A common photometric transformation introduces changes to image brightness. Simple geometric transformations include translations, rotations and skewing (Simard *et al.*, 2003). The learning algorithm can effectively infer the transformation invariance and generalize various plant classes and environmental conditions by introducing these transformations.

The design of a CNN architecture plays an essential role in the discriminative performance of the produced model. Network depth has proven to be of critical importance when designing a CNN architecture (Simonyan and Zisserman, 2014). However, simply stacking more layers leads to problems such as vanishing or exploding gradients (Bengio *et al.*, 1994, Tan and Lim, 2019), which hinder convergence and increase training error. Adding more layers may cause system accuracy to plateau, followed by a rapid drop. Interestingly, this

suggests the increase in training error is not due to overfitting. Therefore, increasing the number of layers can be beneficial when avoiding the aforementioned problems. The CNN architectures of interest are explored in the following subsections.

2.3.1.1 ResNet

ResNet was proposed by He *et al.* (2016). They use a deep residual learning framework in their architecture to combat the degradation issue that arises with high network depth. This is achieved with ‘shortcut connections’ (Ripley, 2007), or residual links, which allows connections to skip one or more layers. These shortcut connections are simple identity mappings where their outputs are added to the outputs of the stacked layers. The system learns residual functions without adding computational complexity making the deep network easier to optimize and faster to compute. A visual representation of the residual block found in the ResNet architecture that uses a shortcut connection is given in Figure 2.5 (He *et al.*, 2016).

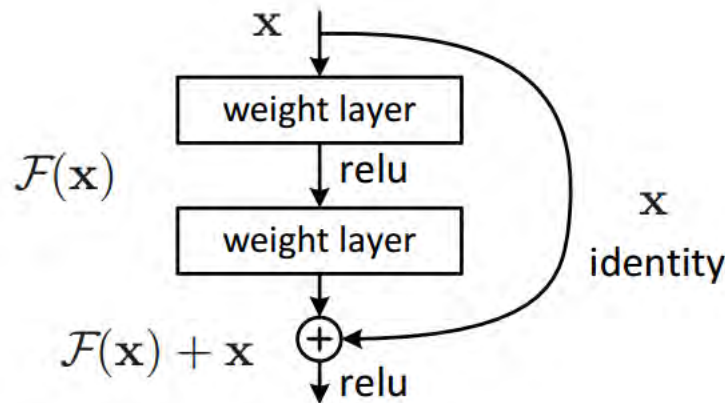


Figure 2.5: The ResNet residual block.

He *et al.* (2016) compared their proposed architecture to popular existing state-of-the-art architectures such as VGG-16, GoogLeNet and PReLU-net. They performed the comparison using the ImageNet 2012 classification dataset (Russakovsky *et al.*, 2015). This image dataset consists of 1000 classes and over 1.4 million images. They used just over 1.2 million images for training, 50k for validation and 100k for testing.

They found ResNet obtained consistently lower top-5 error rates than the other architectures considered. The top-performing existing architecture models obtained an error equal to the poorest performing ResNet model. The ResNet models showed a trend of decreasing error rate with increasing layer depth⁴. The 152 layer ResNet model obtained the lowest error rate of 4.49%.

2.3.1.2 VGG

Simonyan and Zisserman (2014) mitigated the typical issues that arise with very deep networks by utilizing many convolutional layers with small 3×3 filter sizes. The chosen 3×3 filter size is the smallest one that captures the notion of left and right, up and down and centre. The decision to use stacked 3×3 filters instead of the typical larger 5×5 and 11×11 filters were based on the design of the ZfNet (Zeiler and Fergus, 2014) architecture, where they found that small filter sizes often can improve the performance of a CNN as well as lower the computational complexity by reducing the number of parameters.

Simonyan and Zisserman (2014) used Maximum Pooling (maxpool) layers in between the various 3×3 convolutional layers to reduce volume size. The maxpool layer achieves this by using a 2×2 filter to calculate the maximum value of the current feature map. The values in that patch are replaced with the maximum value only, reducing data size. This results in a feature map that highlights the most prevalent feature in each patch. Furthermore, they performed padding to maintain the spatial resolution of the data (Ranzato *et al.*, 2007). A visual representation of the exact architectural design for the VGG architecture of different layer depths is given in Figure 2.6 (Simonyan and Zisserman, 2014).

Simonyan and Zisserman (2014) compare the performance of their proposed architecture to that of other state-of-the-art and well-known architectures such as GoogLeNet and ZfNet on the ILSVRC-2014 dataset. The ILSVRC-2014 dataset is a subset of the ImageNet dataset used in the Large Scale Visual Recognition Challenge 2014 competition (Russakovsky *et al.*, 2015). VGG, GoogLeNet and ZfNet obtained 7.3%, 6.7%

⁴a maximum layer depth of 152 layers was considered.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 2.6: The VGG architectures for 11, 13, 16 and 19 layers.

and 14.8% top-5 test error rates, respectively. The results highlight VGG's competitive performance compared to the 2014-ILSVRC competition winner, GoogLeNet. Overall, although GoogLeNet outperformed VGG, VGG still became popular due to its simplicity, homogenous topology and increased layer depth (Khan *et al.*, 2020).

2.3.1.3 GoogLeNet

GoogLeNet, also known as Inception-V1, was proposed by Szegedy *et al.* (2015) and was the winner of the 2014-ILSVRC competition. Szegedy *et al.* (2015)'s design focuses on computational efficiency and practicality. They counter the vanishing gradient problem by adding auxiliary classifiers connected to the network's middle layers, providing regularization and speeding up the convergence rate. As the layers go deeper, they lose spatial information. By adding the auxiliary classifiers at various stages of the network, this spatial information is retained.

Similar to VGG, GoogLeNet uses maxpool to reduce volume size in the middle layers. However, they use average pooling, followed by fully connected layers directly before each

activation layer for each auxiliary and final classifier.

The majority of the GoogLeNet architecture consists of inception blocks stacked together. The concept behind the inception block is to introduce multi-scale convolutional transformations using the split, transform and merge idea. The block comprises filters of different sizes, namely 1×1 , 3×3 and 5×5 , to capture spatial information at different scales. The split, transform and merge idea addressed the need to learn multiple variations found within the same image class at different resolutions. A visual representation of the inception block described above is given in Figure 2.7.

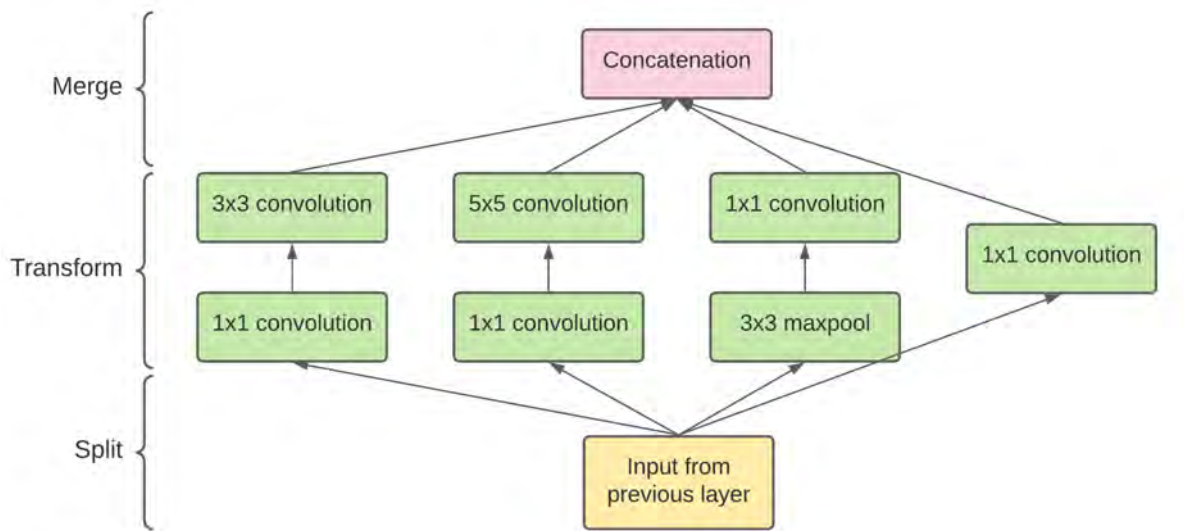


Figure 2.7: The GoogLeNet inception block with the split, transform and merge concept.

Szegedy *et al.* (2015) utilized the ImageNet 2012 classification dataset (Russakovsky *et al.*, 2015) for testing. They achieved a top-5 error rate of 6.67% while VGG, MSRA and other top architectures achieved top-5 error rates of 7.32%, 7.35% and above 11.2%, respectively.

2.3.2 Improving Model Generalization

A model's ability to generalize the data it is trained on for avoiding overfitting has a considerable impact on its final performance. Overfitting occurs when a model learns to fit the training data 'perfectly' and can no longer generalize the problem well (Dietterich, 1995). In the context of this research, an overfitted model does not effectively account

for environmental changes, image background changes, the use of different cameras etc. There are multiple methods employed to avoid overfitting in deep learning. These methods include data augmentation, dropout layers, batch normalization and early stopping.

2.3.2.1 Data Augmentation

Deep learning has proven to perform exceptionally well on computer vision problems. However, it is heavily reliant on large amounts of training data to avoid overfitting ([Halevy et al., 2009](#)). Data augmentation encompasses a range of techniques used to enhance the size and quality of datasets intended for training machine learning models. Some image augmentation algorithms include geometric transformations, photometric augmentations, noise injection, and random erasing ([Shorten and Khoshgoftaar, 2019](#)).

Geometric transformations refer to operations that alter the image relative to the coordinate system. These include rotations, flips, translations, sheers and zooms.

Photometric augmentations refer to operations that alter the pixel colour values within the data. Popular augmentations include isolating a single colour channel or adjusting brightness or lighting in the image.

Correctly selecting suitable augmentation techniques to apply to a dataset is beneficial for avoiding unwanted biases that may be present in the data. However, over-zealous application of augmentation parameters can result in abundant noise, yielding poor classification results.

2.3.2.2 Dropout

Dropout is a regularisation technique first introduced by [Hinton et al. \(2012\)](#). It is a simple technique designed to prevent a model from overfitting. The fundamental concept of dropout is to omit randomly selected neurons during each training iteration of a model. An inputted probability p determines the likelihood of each neuron being omitted from the network in each iteration. Dropout is applied to all layers except the final output

layer. A visual representation of the application of dropout to a neural network of three layers is given in Figure 2.8 (Hinton *et al.*, 2012). The first network is fully connected. The second network has a dropout of 50% applied to it — i.e. its neurons had a 0.5 probability of being dropped per epoch.

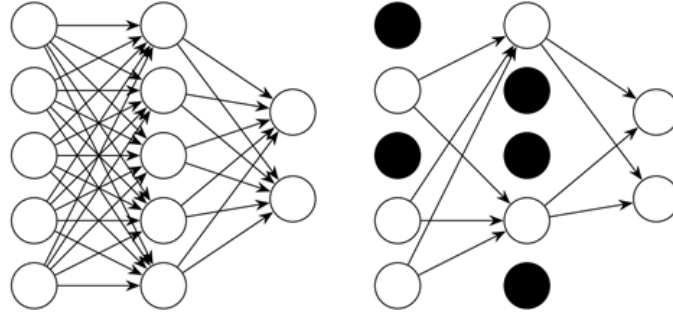


Figure 2.8: The application of 50% dropout to a three-layer network.

The random omission of neurons during training encourages the remaining neurons to reduce reliance on the presence of other neurons. Once training concludes, a whole network of all the neurons is used for the final model. However, each neuron's output is first multiplied by the probability that the neuron was not omitted, i.e. multiplied by $1 - p$. This essentially creates an average of all the possible networks used during training (Labach *et al.*, 2019).

Dropout has been proven to be effective at enabling NNs to train for extended periods without overfitting, typically yielding improved test accuracies (Srivastava *et al.*, 2014).

2.3.2.3 Batch Normalization

Batch normalization (BN) is a technique developed by Ioffe and Szegedy (2015) to allow models to use higher learning rates, increase their speed of convergence, avoid overfitting and minimize the need for dropout layers. BN seeks to achieve this by minimizing the *internal covariate shift* (ICS), the phenomenon where the distribution of layer inputs change during training due to updates to the preceding layers (Shimodaira, 2000). This continual change is thought to impact a model's training negatively.

Ioffe and Szegedy (2015) propose to minimize ICS by introducing additional network layers designed to normalize the layer inputs of intermediate layers. It is widely accepted that normalizing a neural network's input data to have a zero mean and a constant standard deviation is significantly beneficial to the speed of convergence of the neural network (LeCun *et al.*, 1998, Wiesler and Ney, 2011). This idea naturally extends to be applicable across the intermediate layers of a deep neural network. Linearly transforming data to have zero means and constant deviation is known as whitening. Therefore, BN aims to whiten the activations at specific intervals within a network.

Ioffe and Szegedy (2015) applied BN to the state-of-the-art deep learning architecture, GoogLeNet, and found that applying BN allowed the GoogLeNet model to achieve higher accuracies while reaching the same accuracies in 14 fold fewer training steps. The accuracy improvement was attributed to BN increasing the initial learning rate by 30-fold, allowing the model to converge faster and obtain higher accuracy than GoogLeNet alone.

2.3.2.4 Early Stopping

Early stopping is a technique employed to prevent overfitting by terminating model training when given criteria are met (Prechelt, 1998). The typical criteria chosen is the improvement in validation results during training: if the validation result has not improved within the specified number of training iterations, training is stopped prematurely. Using early stopping for model regularization has the advantage of also lowering computational complexity (Raskutti *et al.*, 2014).

2.3.3 Parameter Tuning

Appropriately substantiating the choice of architecture and hyperparameters improves both the accuracy and training speed of deep learning models (Rogachev and Melikhova, 2020). This is shown to be vital in the engineering applications of classification and modelling (Jiménez *et al.*, 2009). Hyperparameters⁵ are the model parameters passed

⁵Primary CNN hyperparameters include learning rate, batch size and number of hidden layers.

prior to training. They dictate the network structure and the adjustments and corrections allowed during the learning process.

One method to find the optimal hyperparameters is to compare the performance of all models trained on a range of combinations and select hyperparameters that produced the top-performing model, typically in terms of accuracy. This optimization is commonly achieved through a grid search. A popular framework often used in hyperparameter optimization is KerasTuner (O'Malley *et al.*, 2019).

2.3.3.1 Grid search

A standard linear grid search is the simplest means to determine optimal parameters (Jiménez *et al.*, 2009). A grid search, also known as a parameter sweep, is an exhaustive search that finds the best combination of input parameters. A basic visual representation of the division of the search area during a grid search is given in Figure 2.9, where the x_1 and x_2 are two parameters being optimized, the mapped crosses show the coordinates of the parameters for each model evaluated for the grid search and the contour lines represent the model performances. The blue contours represent areas of good results, and the red contours represent areas of poor results. The green bars along the top and right edges indicate each distinct value of x_1 and x_2 evaluated.

2.3.3.2 Random Search

In the above approach, the search area was uniformly divided into a grid to cover the maximum number of combinations possible. One method to reduce the computational expense of a grid search is to consider a smaller number of points that are randomly selected (Huang *et al.*, 2007, Bergstra and Bengio, 2012). A visual representation of a search area for a random search is shown in Figure 2.10, where the x_1 and x_2 are two parameters being optimized, the mapped crosses show the coordinates of the parameters for each model evaluated for the random search and the contour lines represent the model performances. The blue contours represent areas of good results, and the red contours

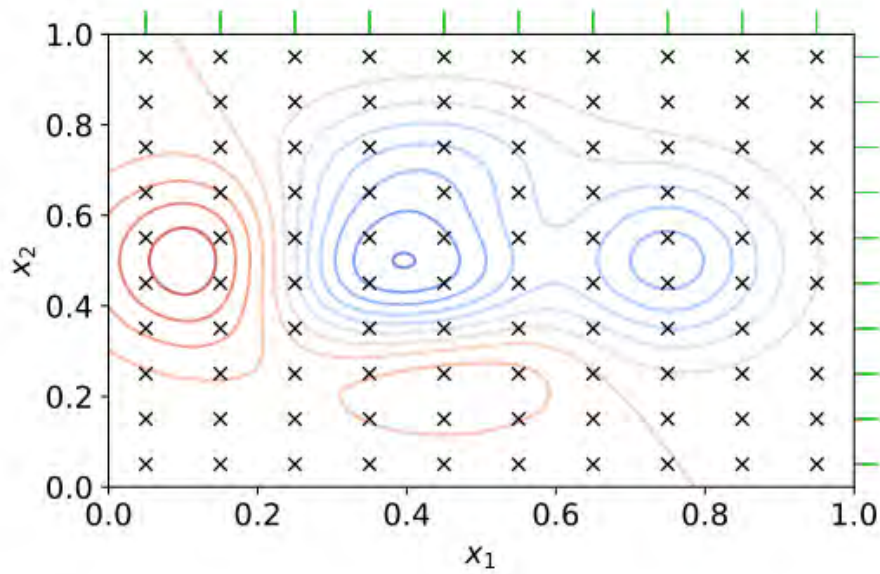


Figure 2.9: Graphical representation of the search area in a grid search [CC BY-SA 4.0].

represent areas of poor results. The green bars along the top and right edges indicate each distinct value of x_1 and x_2 evaluated. Compared to Figure 2.9, there is a notably higher amount of green lines present, indicating that a higher number of individual values are considered for each hyperparameter in a random search than in a grid search.

There are many approaches to choosing the randomly selected parameters. An easy-to-use and popular approach is the Random Search (Bergstra and Bengio, 2012) optimizer provided within the KerasTuner framework (O'Malley *et al.*, 2019).

The KerasTuner framework also contains the Bayesian Optimization (Snoek *et al.*, 2012) and Hyperband (Li *et al.*, 2016) optimizers, both of which are considered more efficient than the Random Search optimizer and are explored below. The advantage of the Hyperband optimizer is that it can find reasonably good configurations in situations of lower computational power or when quick results are required (Falkner *et al.*, 2017). Bayesian Optimization, while it is considered slower, has been shown to outperform many other state-of-the-art optimization algorithms on several challenging benchmark functions, including Hyperband (Jones, 2001).

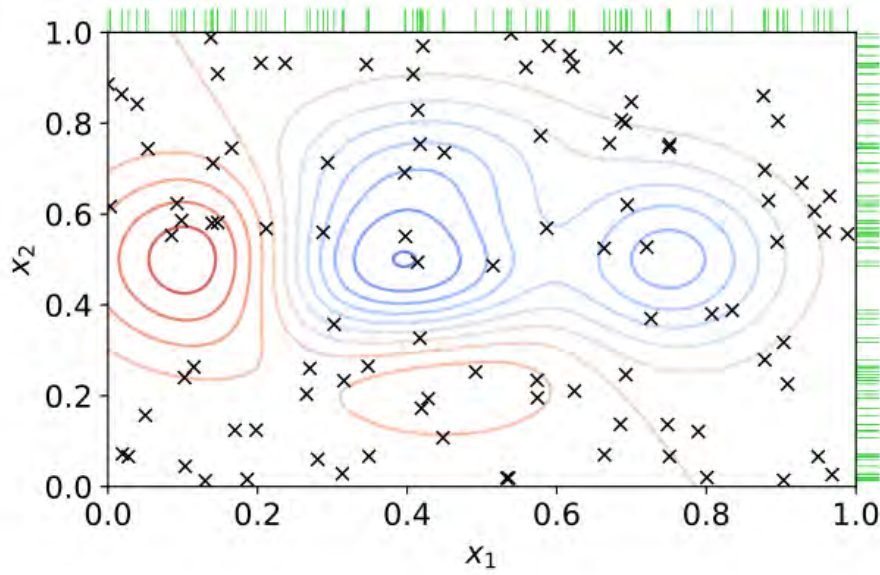


Figure 2.10: Graphical representation of the search area in a random search [CC BY-SA 4.0].

2.3.3.3 Hyperband

Hyperband (Li *et al.*, 2016) is an optimizer based on the Random Search optimizer. It uses early stopping and adaptive resource allocation to decrease the system’s computational expense and increase the tuning speed of the system.

Hyperband seeks to minimize the distance to the optimal solution as quickly as possible. The Hyperband algorithm extends on the Successive Halving algorithm proposed by Jamieson and Talwalkar (2016) and calls it as a subroutine. Fundamentally, the Successive Halving algorithm evaluates the performance of all possible hyperparameter configurations, discards the worst-performing half and repeats until one last configuration remains.

Overall, the Hyperband algorithm can exploit situations where adaptive resource allocation works well (Li *et al.*, 2016).

2.3.3.4 Bayesian Optimization

Snoek *et al.* (2012) based their work on the Bayesian methods for optimization proposed

by [Mockus *et al.* \(1978\)](#), however, adapted it to be applicable for hyperparameter selection for general machine learning algorithms. Their proposed algorithm is known as Bayesian Optimization. Bayesian Optimization relies on predicting the next point to test that will provide the most probable result improvement.

The idea is to not rely on a local gradient or Hessian approximations but rather to consider all information available from the previous evaluations of the model. Therefore, at the cost of performing more computation to determine the next point to evaluate, a procedure is created to find the minimum loss of a model with relatively few evaluations. In situations where the evaluation of the model is computationally expensive, as is the case with training deep learning algorithms, it is somewhat justified to allocate extra computational time to make better decisions.

A visual representation of the points chosen during Bayesian Optimization is given in Figure 2.11, where the x_1 and x_2 are two parameters being optimized, the mapped crosses show the coordinates of the parameters for each model evaluated for the Bayesian Optimization algorithm and the contour lines represent the model performances. The blue contours represent areas of good results, and the red contours represent areas of poor results. The green bars along the top and right edges indicate each distinct value of x_1 and x_2 evaluated.

Due to its ability to weigh up the quality of the contribution of each experiment before it is run, Bayesian Optimization has been shown to obtain better results in fewer evaluations than grid search, random search and Hyperband, especially on large datasets ([Thornton *et al.*, 2013](#)).

[Snoek *et al.* \(2012\)](#) evaluated their proposed optimization algorithm on many challenging optimization benchmarks. They found that their proposed algorithm obtained the lowest test error rate of all reported for the competitive CIFAR-10 ([Krizhevsky *et al.*, 2009](#)) benchmark by 3%.

Therefore, Bayesian Optimisation is considered adequate for this research's applications due to the fast training speed and ability to outperform most other optimization tech-

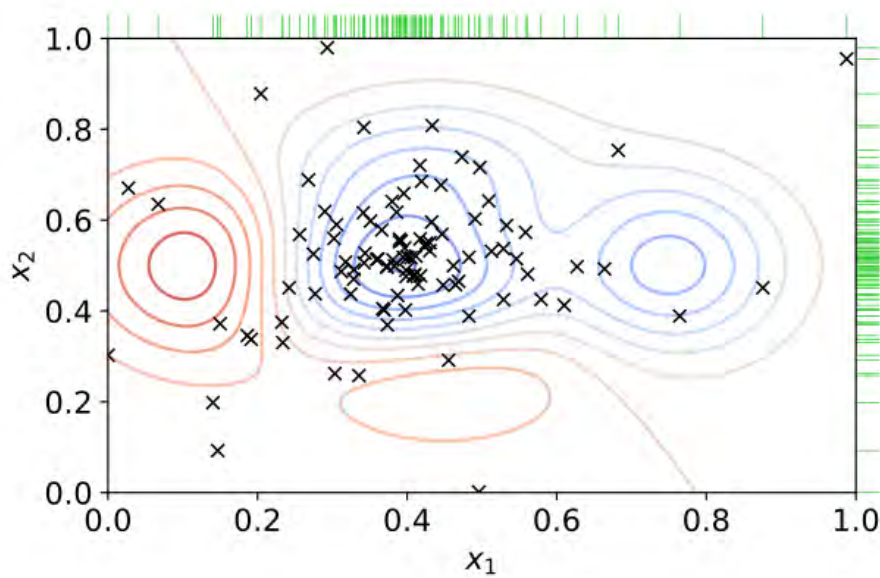


Figure 2.11: Graphical representation of the search area in Bayesian Optimization [CC BY-SA 4.0].

niques, given the same training time. The detail of the design choices for this research is provided in Chapter 3.

2.4 Summary

This chapter analyzed various related studies and expanded on the concepts and techniques used in those studies. The importance of plant stress and disease classification was explored, and the potential for automating early disease and stress detection using spectroscopy and machine learning was introduced.

The spectral reflectance of plant leaves was investigated, and studies pertaining to the wavelengths sensitive to plant stress were compared. Based on the literature, wavelengths in the far-red and NIR ranges were most suitable for general plant stress detection. Existing NIR plant disease datasets were lacking, and thus procedures for collecting new datasets were explored.

Furthermore, machine learning concepts were investigated, and the CNN-based deep learning algorithm was highlighted for potential modelling due to its fast-growing wide

acceptance. The performance and basic design of various CNN architectures, namely ResNet, VGG and GoogLeNet, were analyzed. Their performance was data-dependent. Therefore, further investigation on plant image data was shown to be required before choosing an optimal architecture.

Moreover, parameter tuning was introduced to optimize a machine learning model's performance. This systematic approach is essential as this research proposed collecting a new dataset that is untested on those CNN architectures will be collected. KerasTuner's Bayesian Optimization was selected as the most suitable hyperparameter optimization algorithm for this purpose.

3

Methodology

This chapter presents and discusses the approach used to create an automated early plant stress classification system. The system design choices are explained in the following sections — implementation algorithms and data collection process are outlined.

3.1 Methodology Overview

Training a CNN model for optimal results requires selecting an appropriate architecture. For the success of the proposed research, this implies that the chosen CNN architecture must perform well on plant data specifically. Thus the hyperparameters of the CNN models are extensively tuned to produce optimal neuron weights for plant leaves, separately for visible and multispectral data.

3.1.1 CNN Architecture Selection

Three popular CNN architectures, namely ResNet, GoogLeNet and VGG-16, were considered for the final model. The architecture for the final model was selected by investigating and comparing each architecture's discriminative ability on multiclass plant stress classification problems.

Three models were trained for evaluating the chosen architecture’s performances. The dataset was split into three sets for this process: training, validation, and test data. Each model’s hyperparameters were thus tuned on the validation set and fine-tuned on the combined training and validation set. Finally, the three models were evaluated on the unseen test data to reflect the models’ unbiased performance accurately.

Furthermore, data contamination was avoided during data augmentation as it was applied only to the training data and not to the validation data during tuning or the test data. The specifications of the implementation of this system are given in Section 5.2.1.

3.1.2 Plant Stress Classification System

Based on the objectives given in Section 1.5, an automated plant stress classification system was developed. This system was designed to accurately identify both early and advanced symptoms of plant dehydration across a variety of plant species through the use of multispectral imaging.

It is vital to have a suitably designed dataset structure for the model to be trained on to attain those objectives. The specifications of such a dataset are given in Section 3.2.

The high-level view of the system, shown in Figure 3.1, operates as follows. The model construction process begins with passing the best architecture design, based on preliminary experiments, to the model builder to produce an untrained model. The training data is sent to an augementer to extend the data and introduce variation. The untrained model, augmented data and tunable hyperparameters are input to a Bayesian Optimization tuner. Hence, an optimal model is produced. This model is retrained on the augmented training data with the validation data used to estimate the model’s performance during training. The final trained model is tested on the unseen test data. The specifications of the implementation of this system are given in Section 4.2.

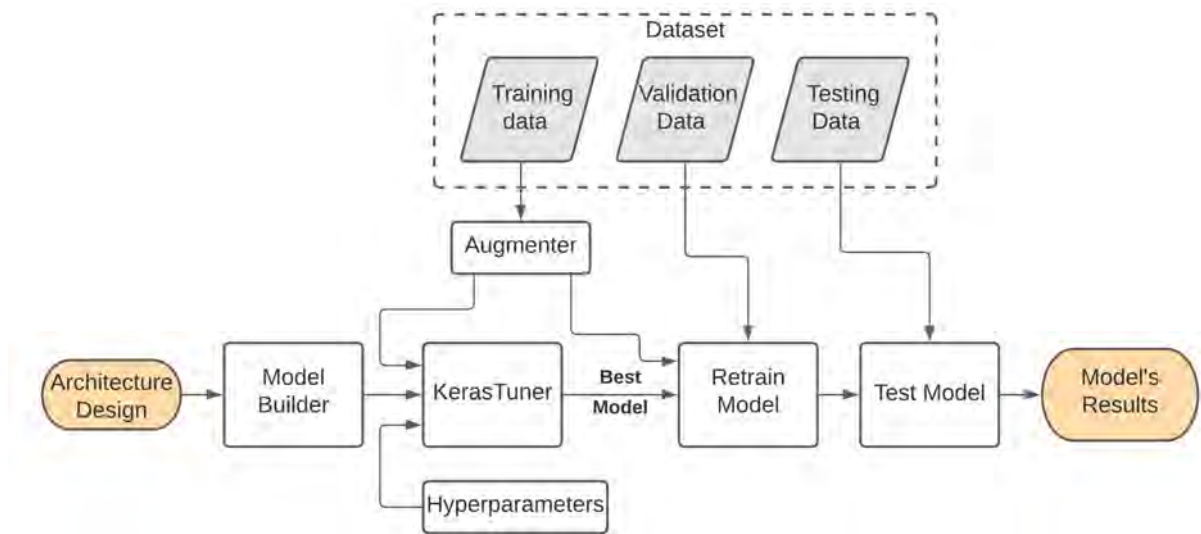


Figure 3.1: High-level view of the early stress detection system.

3.2 Dataset Construction

Based on the objectives in Section 1.5, the datasets collected must be of visible and of multispectral input, must reflect real-world data with complex backgrounds and must be a series of images of a variety of plants species in various states of stress. The structure of the visible and multispectral image datasets should be identical to allow the results to be directly comparable. This facilitates the research question by investigating the effectiveness of multispectral input over visible image input for early stress detection systems.

The selected stress factor chosen for this early stress classification research was water deprivation. Other options included light deprivation and infection with disease. Water deprivation, also referred to as dehydration, was selected due to its ease of application and enforcement as well as simplicity in measuring its progress or degree of dehydration. It was also ideal due to its general applicability to the majority of commonly farmed plant species. Light deprivation was not favoured due to the need for light during data capture and the improbable occurrence of severe light deprivation in the natural agricultural environment. Plant disease is a realistic and prominent plant stress factor in the current agricultural industry. However, in practice, it would require extensive disease inoculation procedures for a large variety of diseases in order to obtain data from a range of plant

species.

Furthermore, the degree of plant disease infection is challenging to monitor prior to visible symptoms appearing on the leaf. Typically visible leaf symptoms are used to determine the severity of infection. However, there are little to no visible symptoms in the early infection stages of a plant. This complicates pinpointing the early infection stage and thus would require specialized equipment and invasive testing to do so. This period prior to the appearance of visible symptoms is the focus of this research. Therefore, a stress that is non-invasively measurable during this period is favoured.

However, a small disease dataset will be collected to validate the system's ability to perform on plant disease data. No *early* disease classification will be performed due to the limitations mentioned above.

The specifics of the data collection performed are given in Section 3.2.1 to 3.2.5.

3.2.1 Mode of VIS-IR Spectroscopy

There are considered three modes of VIS-IR spectroscopy, namely reflectance, interactance and transmittance. The differences between these modes of data acquisition are determined by the position of the light source in relation to the sample being analyzed and the sensor — in this case, the camera. In reflectance mode, the light source is located on the same side of the sample as the camera. In contrast, interactance mode has the light source and the camera in parallel, with the light barrier in between. Finally, transmittance mode uses a light source located on the opposite side of the sample to the camera. A visual representation of all three modes is given in Figure 3.2.

Transmittance mode provides the most efficient and detailed results for data capture (Schaare and Fraser, 2000). However, considering real-world applications, reflectance can be assumed to be the predominant mode of data capture used by those interested in utilizing the final system. This assumption is made due to the following reasons:

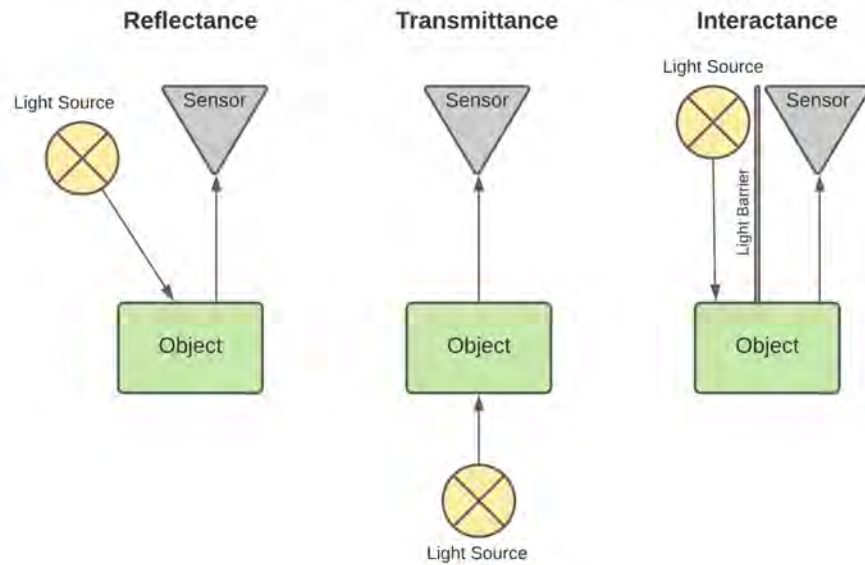


Figure 3.2: Modes of VIS-IR spectroscopy.

- sunlight is the light source most likely to be utilized in the field and thus was used during data capture.
- the majority of the farmable crops used in this research grow low to the ground and thus positioning leaves between the sensor and the sun is not ideal and often not possible.

Therefore, reflectance is the chosen mode of VIS-IR spectroscopy for the data collection performed.

3.2.2 Equipment

The necessary equipment for the required data collection is given in Table 3.1.

The plant types required for collecting the dataset cover a range of representative agricultural plant categories, including leafy greens, fruit or vegetables and herbs. The specific equipment and plants used to satisfy these requirements for this research are given in Section 4.1.1 to 4.1.2.

Table 3.1: Required equipment and its intended uses.

TYPE OF EQUIPMENT	INTENDED USE
Standard visible input camera	Collect images in the visible wavelength range — create visible image dataset
Infrared input camera	Collect images in the VIS-NIR wavelength range — create multispectral image dataset
Soil moisture probe	Collect data on soil moisture content — used to aid in determining the separation of image classes
Moisture meter	Collect data on plant moisture content — used to aid in determining the separation of image classes

3.2.3 Image Composition

To reflect real-world input collected in the field, complex backgrounds of soil, pots, walls, wood, human hands and feet, and other unrelated plants are required. This allows generated machine learning models to generalize the input and learn to ignore background objects that should not affect the final classification.

Furthermore, to introduce sufficient data depth, the following variations, in relation to the focus leaf¹, are required:

- lighting — blocking the light source to different degrees introduces the desired variance in light levels
- rotation — slight rotational changes are introduced by rotating the camera between images.
- positioning — the focus leaf should at minimum partially overlap some part of the centre region of the image with minimal leaf area removed from the image. Slight positioning changes are introduced by small horizontal and vertical shifts of the camera between images.

¹The leaf that the camera is focused on.

- angle — changes in the direction/angle at which the image is captured will introduce angle variance as well as often light and positional variance too.
- zoom — while the lens's zoom is set at a constant zoom, physical zoom is introduced by moving closer/further away from the focus leaf.

These variations are based off of the data capturing approach used by [Hughes and Salathé \(2015\)](#), given in Section [2.2.2.2](#).

3.2.4 Acquisition of Moisture Readings

To monitor the stress levels of the plants, both soil and plant moisture content readings are taken. The soil moisture readings are acquired from between the base roots of the plants. This provides accurate moisture readings for the soil with which the plant's roots interact. The soil moisture probe is inserted 10cm deep. The reading is recorded after the device has settled at a constant value.

For the plant moisture content readings, the two moisture meter probes are rested against the main vein found at the back of each leaf. Once the device displays a constant reading for a minimum of three seconds, an official reading is recorded.

3.2.5 Division of Classes

The images from each species of plant are divided into three classes: *healthy*, *early dehydration* and *advanced dehydration*. The classes represent the degree of stress and are determined by the moisture content readings obtained from the plants photographed. Soil and plant leaf moisture content are both taken into consideration. After a week of sufficient watering and settling into their new environment, the plants are assumed to be healthy and not experiencing symptoms of dehydration on the first day of data collection. The initial readings taken on the first day provide the benchmark healthy readings for

each plant species. Class bins are thus determined using the frequency distribution of the soil and plant moisture readings during the following days of data collection.

The class binning is done by dividing the soil and leaf moisture content readings into bins of determined ranges. The highest and lowest ranges per plant species are used as the *healthy* and *advanced dehydration* bins, respectively. The ambiguous section between those ranges is considered the *early dehydration* bin.

3.3 Summary

This chapter provided a high-level view of the CNN architecture evaluation system for evaluating the performance of ResNet-44, GoogleNet and VGG on plant disease and stress data. Both of which followed the same structural breakdown: model building, hyperparameter tuning, model training and finally, model testing. The resulting early stress classification system was thus designed based on the literature and additional requirements for this study.

Furthermore, the data acquisition techniques for creating a suitable dataset based on the related literature were outlined. Reflectance was selected as the mode of VIS-IR spectroscopy as it was determined to be most applicable for situations where the sun was the chosen light source. The necessary equipment required was listed, and dataset image composition was detailed. The method for the division of the classes in the dataset and procedures for the acquisition of moisture readings for determining the divisions were stated.

4

Implementation

This chapter provides details on the implementation of the systems used in this research. This involves detailing the plant species used to represent the plant types selected in Section 3.2.2, the equipment used to satisfy the equipment requirements also given in Section 3.2.2 and the final binning of the data into appropriate target classes. The system’s infrastructure, including hardware and software, is provided. Finally, the implementation of the architectures, data augmentation, early stopping and parameter tuning are provided in detail.

4.1 Data Collection

The following subsections specify the attributes of the collected dataset, namely plant species, equipment and the division of classes.

4.1.1 Plant Species

The breakdown of the number of plants per species is given in Table 4.1. Six plant species from varieties of the plant types given in Section 3.2.2 are captured during data collection.

Two series of images were captured concurrently for all plant species — one series for visible images and one series for multispectral images. The two image series were designed

Table 4.1: Number of plants per plant species in the collected dataset.

PLANT TYPE	PLANT SPECIES	NUMBER OF PLANTS
Leafy Green	Spinach	23
	Lettuce	13
Fruit/Vegetable	Cauliflower	12
	Tomato	13
	Jalapeno	12
Herb	Rocket	21

to be primarily identical, with input type as the predominant difference. The concurrent capture ensured that the angle, zoom and lighting were consistent between the two series.

For the proof-of-concept multispectral disease dataset collected, spinach was selected as the subject for data collection. Due to the challenges associated with inoculating plants with diseases, spinach was selected for its susceptibility to a wide range of common and easily transmissible diseases.

4.1.2 Equipment

The equipment used to satisfy the requirements listed in Section 3.2.2 is specified in detail as follows. The visible image series was captured using a Canon EOS 700D with an EFS 18-55mm lens set at 35mm zoom. The multispectral series was captured using a converted Canon EOS 800D with the internal visible wavelength filter removed. An EFS 18-55mm lens was set at 35mm zoom with a Kolari Vision K665 58mm external NIR filter. This filter allows far-red light and NIR wavelengths of 665-1000 nm to reach the camera's sensor.

The soil moisture content information was collected using an analogue double-needle moisture probe. The plant moisture content data was collected using a G110 moisture sensor.

4.1.3 Allocation of Classes

As stated in Section 3.2.5, the images from each plant species are divided into three classes: *healthy*, *early dehydration* and *advanced dehydration* using binning. The moisture content

ranges for the bins per plant are given in Table 4.2. The soil readings are given as a moisture rating between 0 and 10.

Table 4.2: Soil and leaf moisture content readings used to separate the data into target classes.

PLANT TYPE	HEALTH STATUS	SOIL MOISTURE CONTENT	PLANT MOISTURE CONTENT
Cauliflower	Healthy	4.5–7.0	1.8–2.2
	Early Dehydration	2.0–4.5	0.4–1.2
	Advanced Dehydration	1.0–1.5	0.0–0.4
Jalapeno	Healthy	7.5–9.0	0.8–1.5
	Early Dehydration	1.5–4.0	0.3–0.8
	Advanced Dehydration	1.0–1.5	0.0–0.3
Lettuce	Healthy	9.5–10.0	1.5–3.4
	Early Dehydration	2.0–6.0	0.6–1.3
	Advanced Dehydration	1.0–1.5	0.2–0.4
Rocket	Healthy	2.5–5.0	1.0–1.3
	Early Dehydration	1.5–2.5	0.4–1.0
	Advanced Dehydration	1.0–1.5	0.0–0.4
Spinach	Healthy	3.0–6.0	0.8–1.6
	Early Dehydration	2.0–0.3	0.3–0.8
	Advanced Dehydration	1.0–1.5	0.0–0.1
Tomato	Healthy	6.5–10.00	1.3–1.8
	Early Dehydration	3.0–6.5	0.9–1.3
	Advanced Dehydration	1.0–2.0	0.4–0.8

These bins divide the dataset into the target classes for training and testing the classification machine learning models. The average pH of the soil used was 6.8.

4.2 Machine Learning Models

The system infrastructure used and implementation details of the machine learning models are covered below.

4.2.1 System Infrastructure

The building and testing of the system’s machine learning models required various hardware and software components as follows.

4.2.1.1 Hardware

A dedicated machine learning server was created for use during this research. The machine learning server components included:

- CPU: AMD Ryzen™ 9 5950X 4.9 GHz 16-core
- RAM: G.Skill Ripjaws V 64GB
- GPU: NVIDIA RTX 3090

4.2.1.2 Software

The following software was used to build, train and test the system:

- Ubuntu 20.04 Linux operating system
- Python 3.9
- TensorFlow 2.6.0
- Keras-Tuner 1.0.4)

4.2.2 Architectures

The chosen architectures in Section 3.1.1, ResNet, VGG and GoogLeNet, are implemented as follows:

4.2.2.1 ResNet-44

Figure 4.1 illustrates the layers that are utilized in the constructed 44-layer ResNet network. This is created using KerasTuner, which selects the number of repeats of the

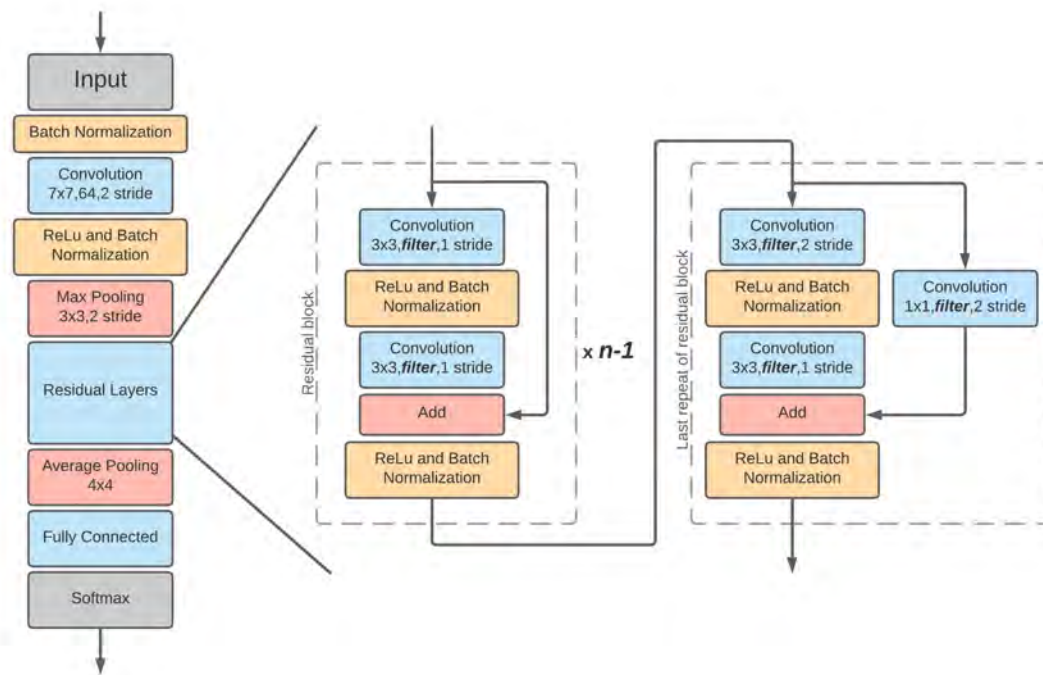


Figure 4.1: Composition of the ResNet-44 network implemented.

Table 4.3: Filter sizes and number of repeats of the residual block in ResNet-44.

	FILTER SIZE (<i>filter</i>)	REPEATS (<i>n</i>)
Block 1	64	2
Block 2	128	5
Block 3	256	5
Block 4	512	2

residual blocks. The filter sizes *filter* and number of repeats *n* of these residual blocks are given in Table 4.3.

The `residualBlock()` function given in Listing 4.1, contains the `reluBatchNormalization()` convenience function that adds a ReLU activation layer followed by a batch normalization layer to a given input. Preliminary experiments on validation data found that this order of these two layers worked better than when they were switched.

```

1  def residualBlock(x: Tensor, downsample: bool, filters: int, kernel_size: int = 3) ->
    Tensor:
2      y = Conv2D(kernel_size=kernel_size,
3                  strides=(1 if not downsample else 2),
4                  filters=filters,
5                  padding="same")(x)
6
7      y = reluBatchNormalization(y)
8
9      y = Conv2D(kernel_size=kernel_size,
10                 strides=1,
11                 filters=filters,
12                 padding="same")(y)
13
14     if downsample:
15         x = Conv2D(kernel_size=1,
16                     strides=2,
17                     filters=filters,
18                     padding="same")(x)
19
20     out = Add()([x, y])
21
22     out = reluBatchNormalization(out)
23
24     return out

```

Listing 4.1: ResNet-44 residual block.

4.2.2.2 VGG-16

Table 4.4 gives the filter sizes and number of convolutions per block for the 16-layer VGG network implemented. All convolutional layers used are set with a kernel size of 3×3 . The first two convolutional blocks are comprised of two convolutional layers followed by one maxpool layer, whereas the last three blocks are comprised of three convolutional layers followed by one maxpool layer.

An example of the code used to construct Block 5 is given in Listing 4.2.

Once all blocks are added to the model, ReLU activation followed by a 50% dropout is applied twice before the final softmax activation layer (Simonyan and Zisserman, 2014).

Table 4.4: Filter sizes of the convolutional blocks in VGG-16.

	NUMBER OF CONVOLUTIONS	Filter size
Block 1	2	64
Block 2	2	128
Block 3	3	256
Block 4	3	512
Block 5	3	512

```

1  model.add(Conv2D(filters=512,
2      kernel_size=(3,3),
3      padding="same",
4      activation="relu"))
5  model.add(Conv2D(filters=512,
6      kernel_size=(3,3),
7      padding="same",
8      activation="relu"))
9  model.add(Conv2D(filters=512,
10     kernel_size=(3,3),
11     padding="same",
12     activation="relu"))
13 model.add(MaxPool2D(pool_size=(2,2),
14     strides=(2,2)))

```

Listing 4.2: Block 5 of VGG-16.

4.2.2.3 GoogLeNet

The original GoogLeNet architecture is implemented for the experiments. The Inception block described in Section 2.3.1.3 is implemented using the function given in Listing 4.3 (Yu *et al.*, 2020).

```

1  def inception_block(x, filters_1x1, filters_3x3_reduce, filters_3x3, filters_5x5_reduce,
2      filters_5x5, filters_pool_proj, name=None):
3
4      conv_1x1 = Conv2D(filters=filters_1x1,
5          kernel_size=(1, 1),
6          padding="same",
7          activation="relu",
8          kernel_initializer=kernel_init,
9          bias_initializer=bias_init)(x)

```

```
10     conv_3x3 = Conv2D(filters=filters_3x3_reduce,
11                       kernel_size=(1, 1),
12                       padding="same",
13                       activation="relu",
14                       kernel_initializer=kernel_init,
15                       bias_initializer=bias_init)(x)
16
17     conv_3x3 = Conv2D(filters=filters_3x3,
18                       kernel_size=(3, 3),
19                       padding="same",
20                       activation="relu",
21                       kernel_initializer=kernel_init,
22                       bias_initializer=bias_init)(conv_3x3)
23
24
25     conv_5x5 = Conv2D(filters=filters_5x5_reduce,
26                       kernel_size=(1, 1),
27                       padding="same",
28                       activation="relu",
29                       kernel_initializer=kernel_init,
30                       bias_initializer=bias_init)(x)
31
32     conv_5x5 = Conv2D(filters=filters_5x5,
33                       kernel_size=(5, 5),
34                       padding="same",
35                       activation="relu", ,
36                       kernel_initializer=kernel_init,
37                       bias_initializer=bias_init)(conv_5x5)
38
39
40     pool_proj = MaxPool2D((3, 3),
41                           strides=(1, 1),
42                           padding="same")(x)
43
44     pool_proj = Conv2D(filters=filters_pool_proj,
45                       kernel_size=(1, 1),
46                       padding="same",
47                       activation="relu", ,
48                       kernel_initializer=kernel_init,
49                       bias_initializer=bias_init)(pool_proj)
50
51     output = concatenate([conv_1x1, conv_3x3, conv_5x5, pool_proj],
52                          axis=3, name=name)
53     return output
```

Listing 4.3: GoogLeNet Inception block.

GoogLeNet provides its output as an array of three activation values. To avoid overfitting, a dropout of 40% is used before the first activation, and a dropout of 70% is used before the last two activations (Szegedy *et al.*, 2015).

4.2.3 Improvement of Model Generalization

Some methods often used to improve model generalization were explored in Section 2.3.2. Both dropout and batch normalization are implemented as layers within the build of the models in Section 4.2.2. Data augmentation and early stopping were implemented as follows:

4.2.3.1 Data Augmentation

Data augmentation is applied two-fold. Firstly it is applied as layers inside the network build. These layers alter the input data before it is fed to the actual model layers. An example of the flip and rotation augmentation layers used is given in Listing 4.4.

```
1 t = keras.layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical")(inputs)
2 t = keras.layers.experimental.preprocessing.RandomRotation(0.2)(t)
```

Listing 4.4: Data augmentation layers.

Data augmentation is also applied to the data directly. This is done using `ImageDataGenerator` from the Keras framework. No augmentation is applied to the data generators for the test and validation data. The augmentations applied to the training data is given in Listing 4.5.

```
1  aug = ImageDataGenerator(  
2      rotation_range=25,  
3      width_shift_range=0.2,  
4      height_shift_range=0.2,  
5      shear_range=0.2,  
6      zoom_range=0.2,  
7      horizontal_flip=True,  
8      vertical_flip=True,  
9      fill_mode="nearest",  
10     brightness_range=[0.2,1.0]  
11 )
```

Listing 4.5: Data augmentation using ImageDataGenerator.

4.2.3.2 Early Stopping

Early stopping is applied at two points in the system: during tuning and training. Both applications of early stopping monitor the validation loss of the model with the aim to minimize it. During tuning, the patience is set to 20, which gives the model 20 epochs to improve on the lowest validation loss achieved before terminating training. During final model training, a patience setting of 250 is used. The code for creating the Keras early stopping callback for training is given in Listing 4.6.

```
1  stop_early = tf.keras.callbacks.EarlyStopping(  
2      monitor="val_loss",  
3      patience=250,  
4      restore_best_weights=True,  
5      mode="min"  
6  )
```

Listing 4.6: Early stopping callback.

4.2.4 Parameter Tuning

The KerasTuner framework with Bayesian Optimization algorithms was used to optimize the models for training. The hyperparameters chosen to explore were optimizer and learning rate. The optimizers known as Adam and SGD were considered along with learning

rates of 10^{-8} to 10^{-4} . The `HyperParameters` object introduces the hyperparameter choices to the architecture's build and passes those choices to the tuner. The tuner object is created and used as shown in Listing 4.7, where `hp` is the `HyperParameters` object used in the model build, `modelBuilder` is the untrained model build, and `stop_early` is the Keras early stopping callback given in Listing 4.6.

```
1  hp = Hyperparameters()
2  hp_opt = hp.Choice("optimizer", values=["Adam", "SGD"])
3  hp_lr = hp.Choice("learning_rate", values=lr_values)
4
5  tuner = kt.BayesianOptimization(
6      hypermodel=modelBuilder,
7      hyperparameters=hp,
8      objective=kt.Objective("val_loss", direction="min"),
9      max_trials=20,
10     overwrite=True
11 )
12
13 tuner.search(
14     x_train,
15     y_train,
16     epochs=100,
17     validation_split=0.2,
18     callbacks=[stop_early],
19     verbose=2,
20     batch_size=32
21 )
22
23 best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]
```

Listing 4.7: Parameter tuning with KerasTuner.

4.2.5 Training and Evaluation

The final model is built and trained using the optimal hyperparameters selected during tuning. The code to train and save the model is given in Listing 4.8 where `best_hps` are the best hyperparameters produced by the code in Listing 4.7, `train_generator` and `val_generator` are the data generators applied to the training and validation data, respectively, and `stop_early` is the Keras early stopping callback given in Listing 4.6.

```
1  bestModel = tuner.hypermodel.build(best_hps)
2
3  history = bestModel.fit(
4      train_generator,
5      epochs=600,
6      validation_data=val_generator,
7      callbacks=[stop_early],
8      verbose=2
9  )
10
11 bestModel.save(filename)
```

Listing 4.8: Model training.

The model is thus evaluated on the training, validation and test data. Where applicable, models with shorter training times were evaluated using a 10 fold evaluation. Listing 4.9 gives the code for loading the saved model and applying model prediction to an input image, where `file` is the file path to the model save and `img` is the input image.

```
1  model = keras.models.load_model(file)
2  output = model.predict(img)
```

Listing 4.9: Model training.

4.3 Summary

This chapter detailed the implementation of the collection and processing of the collected dataset. It provides the six plant species collected, equipment used and division of the class bins for allocation of the target classes.

The system infrastructure, including the hardware and software components, were detailed. The code implementation of the ResNet-44, GoogleNet and VGG architectures were provided, followed by the model tuning and optimization code. The architectures were tuned towards good model generalization such that it was robust to image variations in unseen plant species and plant stress data. This improvement in model generalization was introduced through using dropout, batch normalization, data augmentation and early

stopping. Dropout and batch normalization were implemented within the network layers. Two-fold data augmentation was applied: first as layers inside the network and second as direct augmentation to the data. Lastly, early stopping was applied during tuning and training.

Finally, the code for final model training, storage and inference were provided. The next chapter breaks down and uses these datasets and built models.

5

Experimental Setup

This chapter details the experimental setup used in this research. It begins with the appropriate metric selection followed by the design of all experiments conducted, and finally, the breakdown of all the image datasets utilized in the experiments.

5.1 Measuring Model Performance

Multiple metrics are available for gauging the performance of a given machine learning model. The most commonly used are accuracy, precision, recall, and F-score.

5.1.1 Accuracy

Classification accuracy is the favoured metric for comparing the performance of machine learning models in the majority of plant-related classification research papers available (Ferentinos, 2018, Mohanty *et al.*, 2016, Padol and Yadav, 2016, Sladojevic *et al.*, 2016, Wallelign *et al.*, 2018).

Accuracy is a measurement for the overall correctness of the system and is found by calculating the ratio between the number of correctly classified samples and the total number of input samples:

$$Accuracy = \frac{\text{number of correct classifications}}{\text{total number of classifications attempted}}$$

Accuracy gives a good reflection of the performance of a given model if the test data used is balanced. This means it consists of an equal number of samples belonging to each target class. However, since each test data entry is chosen at random, this is often not guaranteed. The F-score will thus also be reported to cater for random test data.

5.1.2 F-score

The F-score aims to provide an accurate reflection of the performance of a given model. It is calculated as the harmonic mean of the model's precision and recall. Precision is the measure of correctness of all positive classifications per class:

$$Precision = \frac{\text{number of correct classifications of a given class}}{\text{total number of classifications into that class}}$$

or

$$Precision = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

where positive refers to correct classifications.

Contrary to precision, the recall metric counts how many classifications of a particular class were found out of all the samples for that class:

$$Recall = \frac{\text{number of correct classifications of a given class}}{\text{total number of samples for that class}}$$

or

$$Recall = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

where, again, positive refers to correct classifications and negative refers to incorrect classifications.

Finally, the F-score combines the precision and recall of the model by using the following calculation:

$$F - score = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}.$$

As mentioned, both precision and recall, and therefore, F-score, are calculated per class. To get a single F-score for the entire model, the scores for each class need to be combined. The chosen manner in which to combine these values is known as macro averaging. This is where the individual F-scores calculated per class are aggregated together. Unlike with weighted averaging, the F-scores are not weighted before aggregation. It is unnecessary to consider class weights when the target classes in the training and test sets are already correctly balanced.

5.1.3 McNemar Test

McNemar’s test ([McNemar, 1947](#)) is used to determine if there are differences in a dichotomous dependent variable between two related groups. With regards to machine learning, McNemar’s test is used to show statistical significance in the difference of errors produced by two binary models, and thus, highlights if one model significantly outperforms the other.

McNemar’s test is conducted using a contingency table of the classification distribution produced by the two models. An example of such a table is shown in [Table 5.1](#). McNemar’s test aims to signify the difference in errors produced by the two models, and thus, only uses the values b and c from the contingency table.

Table 5.1: Example contingency table for McNemar’s test.

Model 1	Model 2	
	<i>Correct</i>	<i>Incorrect</i>
<i>Correct</i>	a	b
<i>Incorrect</i>	c	d

Two statistics are produced by McNemar’s test, namely χ^2 and *p-value*. They are calculated using the following equations:

$$\chi^2 = \frac{(b - c)^2}{(b + c)}$$

and

$$p\text{-value} = 1 - \text{cdf}(\chi^2)$$

where *cdf* is the cumulative distribution function of χ^2 . These equations assume that the accumulative value of *b* and *c* are large enough to use the χ^2 distribution with 1 degree of freedom to approximate the distribution of the test statistic.

However, if $b + c < 25$, this approximation fails and an exact *p-value* must rather be calculated using a binomial distribution. This is done with the following equation:

$$\text{exact-}p\text{-value} = 2 \sum_{i=b}^{b+c} \binom{b+c}{i} 0.5^{b+c}.$$

For *p-values* lower than the significance threshold of $\alpha = 0.05$, the difference in the performance of the tested models is considered significant, and thus, *the null hypothesis is rejected*.

5.2 Test Models

The following test models were created to satisfy the research objectives in Section 1.5. First, the models used to evaluate the chosen architectures are provided, followed by the models used to evaluate the effectiveness of multispectral input over visible input.

5.2.1 CNN Architecture Evaluation

To select the most appropriate CNN architecture out of ResNet-44, GoogLeNet, and VGG-16, six models were trained for the respective experiments listed below:

- Experiment 1.1 — Compares the performance of the architectures when trained and tuned on the balanced Mendeley Leaves dataset.
- Experiment 1.2 — Compares the performance of the architectures when trained and tuned on the balanced PlantVillage dataset.
- Experiment 1.3 — Compares the performance of the architectures when trained and tuned on the balanced collected visible and multispectral datasets.

The results of these experiments are presented in Section 6.1.

5.2.2 Early Stress Detection

Several models were generated for the comparative testing of the proposed system on visible and multispectral input for various plant species. All models generated for these experiments use the ResNet-44 architecture.

5.2.2.1 Individual Plant Species

- Experiment 2.1 — Compares the performance of the system for multiclass classification between *healthy*, *early dehydration* and *advanced dehydration* on the various plant species separately.
- Experiment 2.2 — Compares the performance of the system for binary classification between *healthy* and *advanced dehydration* on the various plant species separately.
- Experiment 2.3 — Compares the performance of the system for binary classification between *healthy* and *early dehydration* on the various plant species separately.

5.2.2.2 All Plant Species

- Experiment 3.1 — Compares the performance of the system for multiclass classification between *healthy*, *early dehydration* and *advanced dehydration* on all plant species.
- Experiment 3.2 — Compares the performance of the system for binary classification between *healthy* and *advanced dehydration* on all plant species.
- Experiment 3.3 — Compares the performance of the system for binary classification between *healthy* and *early dehydration* all plant species.
- Experiment 4.1 — Compares the performance of the system for multiclass classification of both the *plant species* and their respective state of dehydration - *healthy*, *early dehydration* and *advanced dehydration*.
- Experiment 4.2 — Compares the performance of the system for multiclass classification of both the *plant species* and their respective state of either *healthy* or *advanced dehydration*.
- Experiment 4.3 — Compares the performance of the system for multiclass classification of both the *plant species* and their respective state of either *healthy* or *early dehydration*.

5.3 Datasets

The datasets acquired for this research are detailed in Section 5.3.1 to 5.3.2. All images are converted to the .png format and resized to 256×256 . For all experiments, a 70:10:20 split on the data was used for training, validation, and testing.

5.3.1 Publicly Available Datasets

Two publicly available plant disease image datasets were acquired for Experiment 1.1 and 1.2. They are detailed below.

5.3.1.1 Mendeley Leaves Dataset

The Mendeley Leaves dataset (Chouhan *et al.*, 2019) consists of over 4500 images of 10 plant species with two target classes per plant species: diseased and healthy. They do not specify which diseases they are. The entire dataset is listed by sample size in Table 5.2.

Table 5.2: Breakdown of the Mendeley Leaves image dataset.

PLANT	HEALTH STATUS	NUMBER OF IMAGES
Alstonia Scholaris	Healthy	179
	Diseased	254
		433
Arjun	Healthy	220
	Diseased	232
		452
Bael	Diseased	118
		118
Basil	Healthy	148
		148

PLANT	HEALTH STATUS	NUMBER OF IMAGES
Chinar	Healthy	103
	Diseased	120
		223
Guava	Healthy	277
	Diseased	142
		419
Jamun	Healthy	279
	Diseased	345
		624
Jatropha	Healthy	133
	Diseased	124
		257
Lemon	Healthy	159
	Diseased	77
		236
Mango	Healthy	170
	Diseased	265
		435
Pomegranate	Healthy	287
	Diseased	272
		559
Pongamia Pinnata	Healthy	322
	Diseased	276
		598
TOTAL		4502

The entire dataset was comprised of images of a single leaf with a dark grey background. As seen in Table 5.2, the Mendeley Leaves dataset also contained varied sample sizes, and thus, required balancing.

Balancing the Mendeley Leaves dataset:

The dataset was balanced by shuffling and splitting using a random number generator with a seed. This ensured that all test data across all experiments using this dataset were unseen and balanced. The smallest target class was *Lemon-Diseased* with 77 images. Therefore, all target classes were limited to 77 samples during balancing, resulting in a balanced dataset of 1540 images.

5.3.1.2 PlantVillage Dataset

The PlantVillage dataset ([Hughes and Salathé, 2015](#)) consists of over 54 000 images of 20 diseases across 14 plant species and are listed by sample size in Table 5.3.

Table 5.3: Breakdown of the PlantVillage image dataset.

PLANT	DISEASE	NUMBER OF IMAGES
Apple	Healthy	1645
	Apple Scab	630
	Black Rot	621
	Cedar Apple Rust	275
		3171
Bell Pepper	Healthy	1478
	Bacterial Spot	997
		2475
Blueberry	Healthy	1502
		1502
Cherry	Healthy	854
	Powdery Mildew	1052
		1906

PLANT	DISEASE	NUMBER OF IMAGES
Corn	Healthy	1162
	Cercospora Leaf Spot	513
	Common Rust	1192
	Northern Leaf Blight	985
		3852
Grape	Healthy	423
	Black Rot	1180
	Esca (Black Measles)	1366
	Isariopsis Leaf Spot (Leaf Blight)	1076
		4045
Orange	Haunglongbing (Citrus Greening)	5507
		5507
Peach	Healthy	360
	Bacterial Spot	2297
		2657
Potato	Healthy	152
	Early Blight	1000
	Late Blight	1000
		2152
Raspberry	Healthy	371
		371
Soybean	Healthy	5090
		5090
Squash	Powdery Mildew	1835
		1835
Strawberry	Healthy	456
	Leaf Scorch	1109
		1565

PLANT	DISEASE	NUMBER OF IMAGES
Tomato	Healthy	1591
	Bacterial Spot	2127
	Early Blight	1000
	Late Blight	1909
	Leaf Mold	952
	Septoria Leaf Spot	1771
	Two Spotted Spider Mite	1676
	Target Spot	1404
	Tomato Mosaic Virus	373
	Tomato Yellow Leaf Curl Virus	5357
		18160
TOTAL		54288

The majority of the dataset was comprised of images of a single leaf with a monotone background created by holding a purple/light piece of paper behind the leaf. However, several also contained other leaves of the same species in the background. These images were not removed as they should not negatively affect the final classification.

Balancing the PlantVillage dataset:

As seen in Table 5.3, the PlantVillage dataset contained varied sample sizes for each target class and thus required balancing to achieve realistic results. The dataset was balanced using the same procedure as done with the Mendeley Leaves dataset. The smallest target class was *Potato-Healthy*, with 152 images. Therefore all target classes were limited to 152 samples during balancing resulting in a balanced dataset of 5776 images.

5.3.2 Collected Datasets

Two image datasets were collected for this research. The first was a simple spinach disease dataset to be used for CNN architecture validation and evaluation of the system's performance on disease data. The second was an early to advanced stress dataset to be used for evaluating the effect of visible versus multispectral input and the viability of an early stress detection system.

5.3.2.1 Disease Dataset

The spinach disease dataset consists of over 400 images. There are over 200 multispectral images and 200 visible images of spinach leaves in *healthy* and *diseased* states. The entire dataset is listed by sample size in Table 5.4.

Table 5.4: Breakdown of the collected spinach disease image dataset.

DATASET	HEALTH STATUS	NUMBER OF IMAGES
Visible	Healthy	114
	Diseased	105
		219
Multispectral	Healthy	115
	Diseased	121
		236
TOTAL		455

The dataset was comprised of images with complex backgrounds. As shown in Table 5.4, the dataset contains varied sample sizes, and thus, requires balancing. The dataset was balanced using the same procedure as when balancing the PlantVillage and Mendeley Leaves datasets. The least represented target class was *Visible-Diseased* with 105 images. Therefore all target images were limited to 105 samples during balancing, which resulted in a dataset of 420 images.

5.3.2.2 Early and Advanced Stress Dataset

The collected early and advanced stress dataset consists of over 2400 images. There are over 1200 multispectral images and 1200 visible images of six plant species with three target classes per plant species: *healthy*, *early dehydration*, and *advanced dehydration*. The entire visible dataset is listed by sample size in Table 5.5. The multispectral dataset replicates the structure of the visible dataset, and thus, has an identical breakdown of sample size.

Table 5.5: Breakdown of the collected early and advanced dehydration image dataset.

PLANT	HEALTH STATUS	NUMBER OF IMAGES
Cauliflower	Healthy	74
	Early Dehydration	159
	Advanced Dehydration	90
		323
Jalapeno	Healthy	52
	Early Dehydration	57
	Advanced Dehydration	57
		166
Lettuce	Healthy	54
	Early Dehydration	69
	Advanced Dehydration	54
		177
Rocket	Healthy	53
	Early Dehydration	75
	Advanced Dehydration	53
		181
Spinach	Healthy	69
	Early Dehydration	52
	Advanced Dehydration	52
		173

PLANT	HEALTH STATUS	NUMBER OF IMAGES
Tomato	Healthy	55
	Early Dehydration	83
	Advanced Dehydration	56
		194
TOTAL		1214

The dataset was composed of images with complex backgrounds that used natural sunlight as the light source. Samples of the images of the various plant species from the visible and multispectral datasets collected are given in Figure 5.1.

As shown in Table 5.5, the dataset contains varied sample sizes, and thus, requires balancing. The dataset was balanced using the same procedure as when balancing the PlantVillage and Mendeley Leaves datasets. The least represented target classes were *Jalapeno-Healthy*, *Spinach-Early Dehydration* and *Spinach-Advanced Dehydration*, each with 52 images. Therefore, all target classes were limited to 52 samples during balancing, which resulted in a dataset of 936 images.



Figure 5.1: Sample images of each plant species in the collected dataset.

5.4 Summary

This chapter detailed the metrics and statistic test chosen for model performance measurement, namely accuracy, F-score and McNemar test. It listed the three CNN architecture evaluation experiments and nine visible and multispectral stress detection experiments for selecting an architecture and evaluating the difference of visible and multispectral input performance. It provided the composition and balancing of the publicly available image datasets used and the visible and multispectral datasets collected. The next chapter makes use of this experimental setup.

6

Results and Discussion

This chapter presents and discusses the results obtained by the tests conducted for the experiments detailed in Section 5.2.

6.1 Evaluation of CNN Architecture

The results of the parameter tuning performed to prepare for the CNN selection experiments, the experiments and the conclusion reached after analysis are given below.

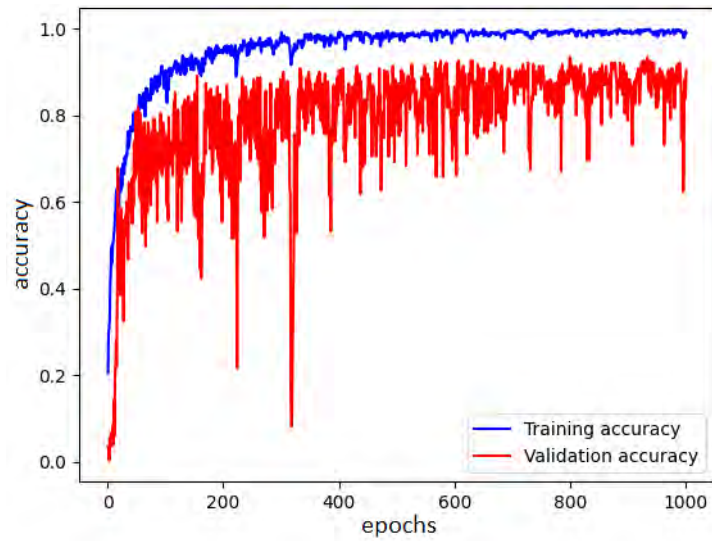
6.1.1 Parameter Tuning

Parameter tuning was carried out over 25 trials to find the best learning rate per model design. The search space included learning rates of $1e^{-2}$ to $1e^{-6}$. A maximum of 1000 epochs were used to find the best epoch. The parameters that produced the *optimal* models per dataset-architecture pairing are given in Table 6.1.

The training and validation accuracies for the ResNet-44 and GoogLeNet models produced on the Mendeley Leaves dataset with the parameter selected in Table 6.1 are graphed in Figure 6.1 and 6.2, respectively. These graphs show that ResNet-44 creates a more erratic learning curve than GoogLeNet. This is attributed to the higher learning rate selected for ResNet-44. However, ResNet-44 achieved higher accuracies at a far earlier epoch than GoogLeNet and obtained a higher maximum validation accuracy overall.

Table 6.1: Best parameters determined with KerasTuner for evaluation of ResNet-44, GoogLeNet, and VGG-16 architectures.

Dataset	Architecture	Best Parameters	
		<i>Learning Rate</i>	<i>Epoch</i>
Mendeley Leaves	ResNet-44	$1e^{-3}$	799
	GoogLeNet	$1e^{-4}$	480
	VGG-16	$1e^{-4}$	804
PlantVillage	ResNet-44	$1e^{-4}$	644
	GoogLeNet	$1e^{-4}$	647
	VGG-16	$1e^{-5}$	984
Collected Visible	ResNet-44	$1e^{-4}$	15
	GoogLeNet	$1e^{-4}$	57
	VGG-16	$1e^{-4}$	5
Collected Multispectral	ResNet-44	$1e^{-4}$	22
	GoogLeNet	$1e^{-4}$	20
	VGG-16	$1e^{-4}$	6

**Figure 6.1:** Training and validation accuracy of the ResNet-44 model trained on Mendeley Leaves dataset.

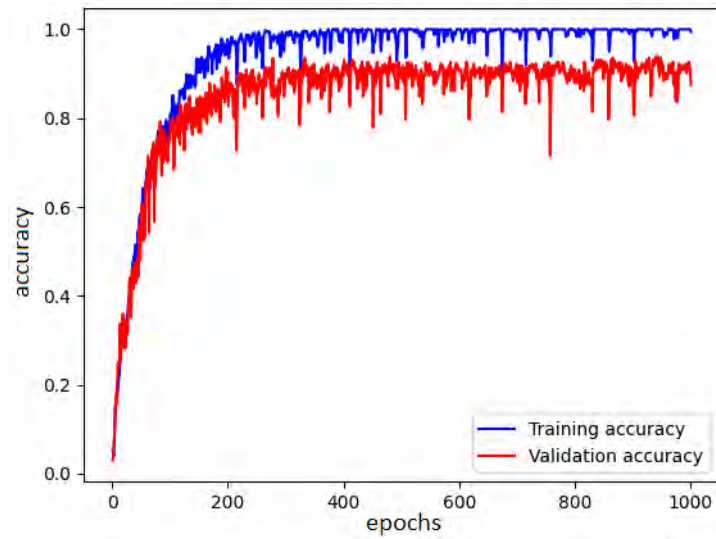


Figure 6.2: Training and validation accuracy of the GoogLeNet model trained on Mendeley Leaves dataset.

It should be noted that ResNet-44 took significantly longer to train than GoogLeNet. However, a plant disease and stress classification system does not require real-time training, and thus, a slow training time does not affect its deployability in the field.

The parameter tuning was performed from scratch to produce an optimized model for the data it was trained on. The models produced were not fine-tuned on a pre-trained model but rather constructed from scratch.

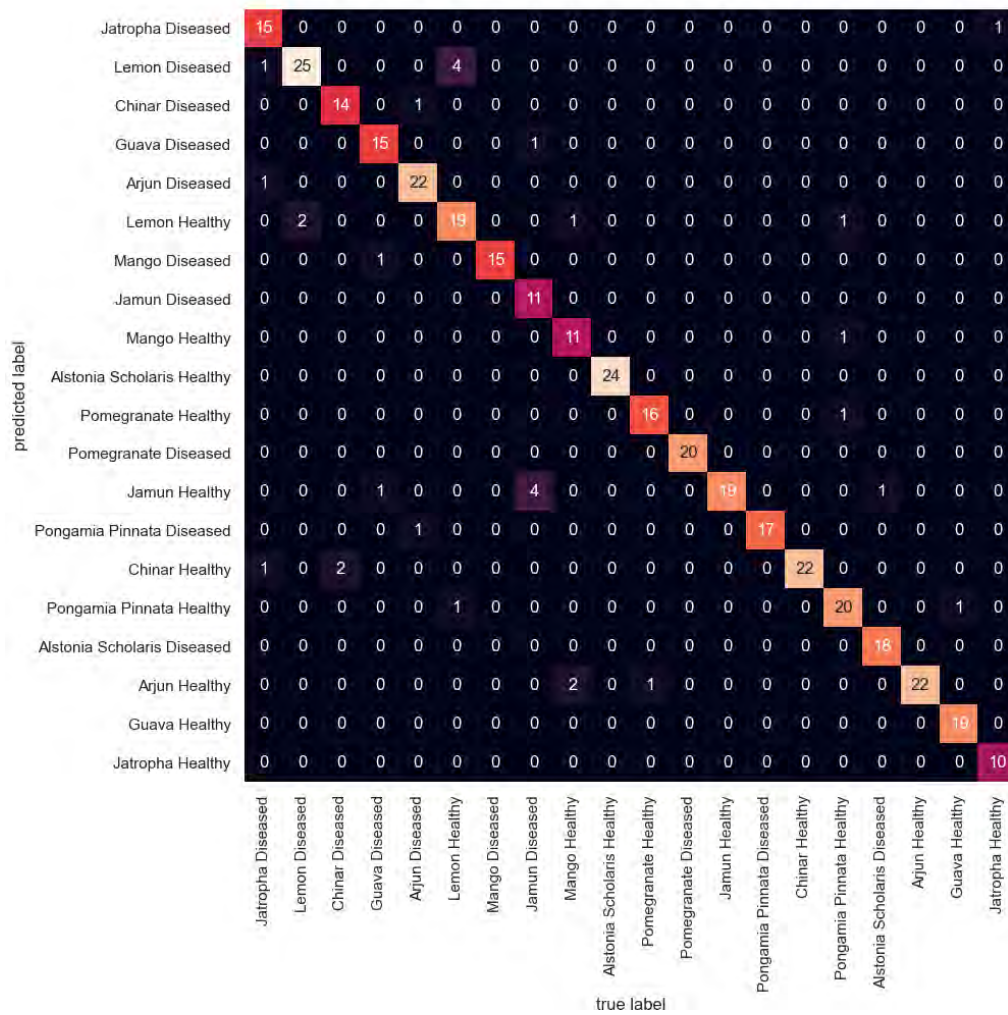
6.1.2 Experiment 1.1

Experiment 1.1 trains and tests the optimal models produced for the ResNet-44, GoogLeNet, and VGG-16 architectures on the balanced Mendeley Leaves dataset. The results given in Table 6.2 show that ResNet-44 obtains the best test accuracy of 91.95%, with GoogLeNet obtaining 2.60% lower and VGG-16 only 0.52% lower than GoogLeNet.

By inspection of the confusion matrix produced by ResNet-44 and given in Figure 6.3, two significant confusions are noted between *Lemon Diseased* and *Lemon Healthy*, and *Jamun Diseased* and *Jamun Healthy*. However, these confusions inevitably highlight the architecture's strong ability to classify plant species, irrespective of the disease, as the correct plant species are still identified in both cases.

Table 6.2: Performance of architectures on balanced Mendeley Leaves dataset.

Architecture	Training		Test	
	<i>Loss</i>	<i>Accuracy</i>	<i>Loss</i>	<i>Accuracy</i>
ResNet-44	0.0123	99.57%	0.3620	91.95%
GoogLeNet	0.0041	100.00%	1.799	89.35%
VGG-16	0.0002	100.00%	0.5885	88.83%

**Figure 6.3:** Confusion matrix of the ResNet-44 model trained and tested on the Mendeley Leaves dataset. Lighter colors indicate higher concentrations of test predictions with correct predictions found along the diagonal.

6.1.3 Experiment 1.2

Experiment 1.2 trains and tests the optimal models produced for the ResNet-44, GoogLeNet, and VGG-16 architectures on the balanced PlantVillage dataset. As with Experiment 1.1, the results in Table 6.3 show that ResNet-44 obtains the top performance with an accuracy of 95.80%. GoogLeNet’s accuracy is only 1.21% lower than ResNet-44’s. VGG-16’s performance is much lower, with an 8.46% drop from ResNet-44’s.

Table 6.3: Performance of the architectures on the balanced PlantVillage dataset.

Architecture	Training		Validation	
	<i>Loss</i>	<i>Accuracy</i>	<i>Loss</i>	<i>Accuracy</i>
ResNet-44	0.0091	99.74%	0.2661	95.80%
GoogLeNet	0.0001	100.00%	0.9716	94.59%
VGG-16	0.0350	99.00%	0.8479	87.34%

Compared to Experiment 1.1, ResNet-44 and GoogLeNet’s accuracy increased while VGG-16’s accuracy decreased. The Mendeley Leaves balanced dataset contains 77 image samples per target class, while the PlantVillage dataset has almost double that with 152 images per target class. This suggests that the additional data aided model training, as both ResNet-44 and GoogLeNet yielded a 4% increase.

However, the PlantVillage dataset has 38 target classes compared to the 20 of the Mendeley Leaves dataset. The almost double number of target classes could account for VGG-16’s performance decrease between the two experiments. VGG-16’s sensitivity to the number of target classes is unwanted as it scales poorly for a plant disease classification system. Therefore, VGG-16 is empirically shown as a non-ideal architecture for plant disease identification as it consistently obtained lower accuracy than ResNet-44 and GoogLeNet and is susceptible to significant performance loss based on changes in the number of target classes.

ResNet-44 and GoogLeNet both obtain the highest accuracies with insignificant differences between the two. With accuracy as the sole metric for comparison, ResNet-44 is considered

the top performer. However, if the computational expense is considered, it may not be feasible with limited resources.

6.1.4 Experiment 1.3

Experiment 1.3 trains and tests the optimal models produced for the ResNet-44, GoogLeNet, and VGG-16 architectures on the balanced collected spinach disease dataset of visible and multispectral input. This small dataset aims to validate the use of multispectral input for diseased data. Due to only one plant species being collected, the results will not accurately represent the performance of a full system.

Due to the small size of the dataset used, 10 repeats of each test is conducted on this data with the results given in Table 6.4. The results obtained are notably high due to the simple nature of the models produced: there are only two target classes. Unlike Experiment 1.1 and 1.2, no classification of plant species is required as a single plant species is used. However, these results indicate that multispectral input can improve the overall accuracy in the case of all three architectures. It is interesting to note that unlike PlantVillage and Mendeley Leaves, the collected disease stress dataset contains complex backgrounds. Therefore, these high accuracies suggest that the system effectively ignores unnecessary information in the input, and thus, can be effectively applied in the field with minimal preprocessing.

Table 6.4: Performance of architectures on the collected spinach disease dataset.

Data	Architecture	Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)
Visible	ResNet-44	57.70±4.32	62.27±8.64	66.82±8.39
	GoogLeNet	94.87±14.74	94.55±14.88	91.82±14.35
	VGG-16	92.37±4.21	96.36±2.08	89.55±3.55
Multispectral	ResNet-44	98.09±1.19	97.95±2.37	100.00±0.00
	GoogLeNet	99.01±0.44	100.00±0.00	100.00±0.00
	VGG-16	99.01±0.79	99.77±0.68	100.00±0.00

6.1.5 Summary of Experiment 1

Overall, all three state-of-the-art architectures reviewed performed well on the plant disease classification datasets used. ResNet-44 and GoogLeNet were constant top performers, with VGG-16's accuracy being slightly lower and noticeably worse when the number of target classes increased.

Both ResNet-44 and GoogLeNet's performances increased when the dataset's size increased, suggesting they could obtain even higher accuracy with an even larger dataset.

ResNet-44 obtained the best overall results of 91.95%, 95.80%, 97.73% and 100.00% on the four images datasets. However, it was noted that ResNet-44 was very computationally expensive. GoogLeNet performed almost as well with 89.35%, 94.59%, 100% and 100% accuracies on the four image datasets and appeared to be much less computationally expensive. ResNet-44 outperformed GoogLeNet on all complicated datasets with large numbers of target classes. GoogLeNet only matched or outperformed ResNet-44 on the simple binary datasets. The two architectures are excellent choices for plant diseases classification systems of both visible and multispectral input, however, ResNet-44 performs better on complex classification tasks.

6.2 Early Water Stress Detection

Results and analysis of the nine experiments conducted on the early water stress detection system follow below. The tested systems make use of the ResNet-44 architecture.

6.2.1 Individual Plant Species

All experiments conducted below train separate models for each plant species.

6.2.1.1 Experiment 2.1

Experiment 2.1 trains and tests the individual models on *healthy*, *early* or *advanced* stages of dehydration per plant species for visible and multispectral input.

While the resulting accuracies given in Figure 6.4 are relatively low, averaging 64.45%. The general trend of multispectral input outperforming visible input is evident. The overall low accuracies obtained are attributed to the small number of training samples used during training.

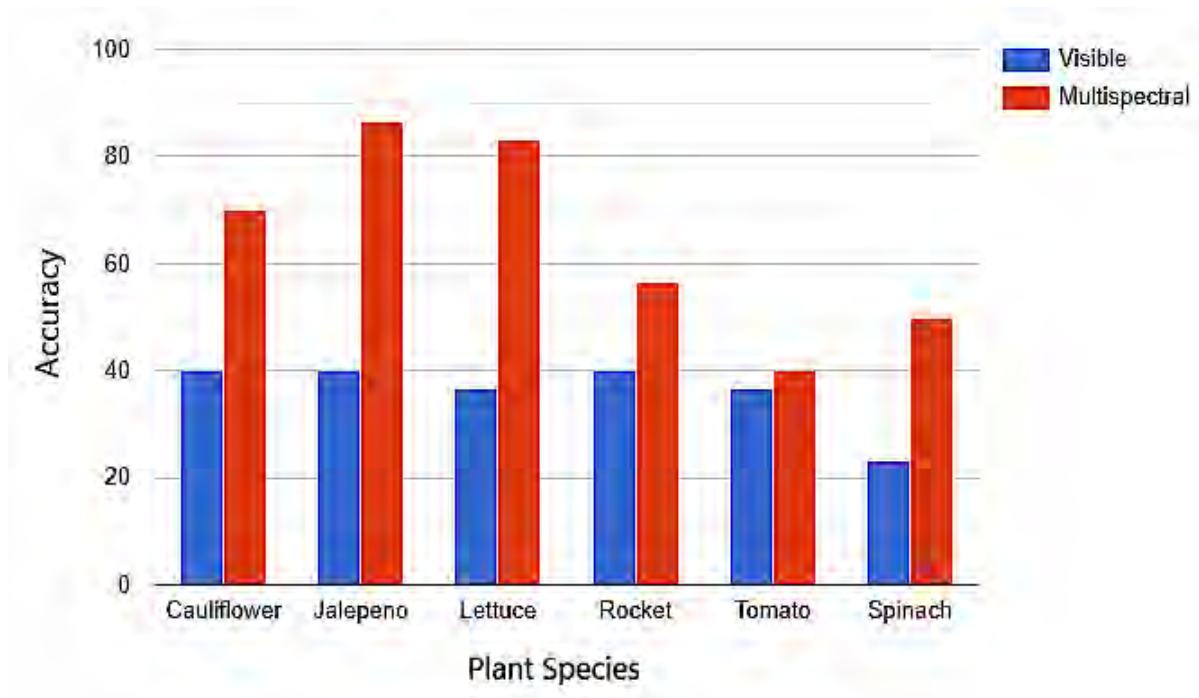


Figure 6.4: Accuracies of the multispectral and visible systems classifying *healthy*, *early dehydration* and *advanced dehydration* on separate plant species.

Jalapeno and lettuce classifiers benefit the most from multispectral input with a 46.67% and 46.66% higher accuracy than the relative accuracies with visible input, respectively.

Conversely, multispectral input enables only the smallest improvement in rocket and tomato. Visual inspection of the rocket and tomato images indicates that the small and thin nature of the leaf shape provides the smallest surface area in the input images by a substantial margin. This small surface area reduces even more in size as dehydration, and thus, leaf curling becomes more advanced. As noted in Section 2.2.1, multispectral

input is intended to distinguish slight changes in reflectance of the leaves and so with a small surface area to consider, the multispectral system will provide limited benefit.

While spinach yields a significant improvement using multispectral input, its overall accuracy is low. The confusion matrix produced by spinach and given in Figure 6.5 shows that the most significant confusion occurs between the *early dehydration* and *advanced dehydration* classes. This suggests that the system can detect dehydration stress but fails to identify the correct stage of the stress.



Figure 6.5: Confusion matrix produced by the multispectral spinach model in Experiment 2.1. Lighter colors indicate higher concentrations of test predictions with correct predictions found along the diagonal.

To analyze the system’s water stress detection performance at different stress stages, *advanced dehydration* and *early dehydration* are evaluated separately in Experiment 2.2 and 2.3, respectively.

6.2.1.2 Experiment 2.2

Experiment 2.2 trains and tests the individual models on *healthy* or *advanced* stages of dehydration per plant species for visible and multispectral input. Figure 6.6 shows that, as with Experiment 2.1, multispectral input consistently outperforms visible input with the

jalapeno and lettuce multispectral classes both obtaining 100% accuracies. The average accuracy for visible input is 70.00% and multispectral input is 84.17%.

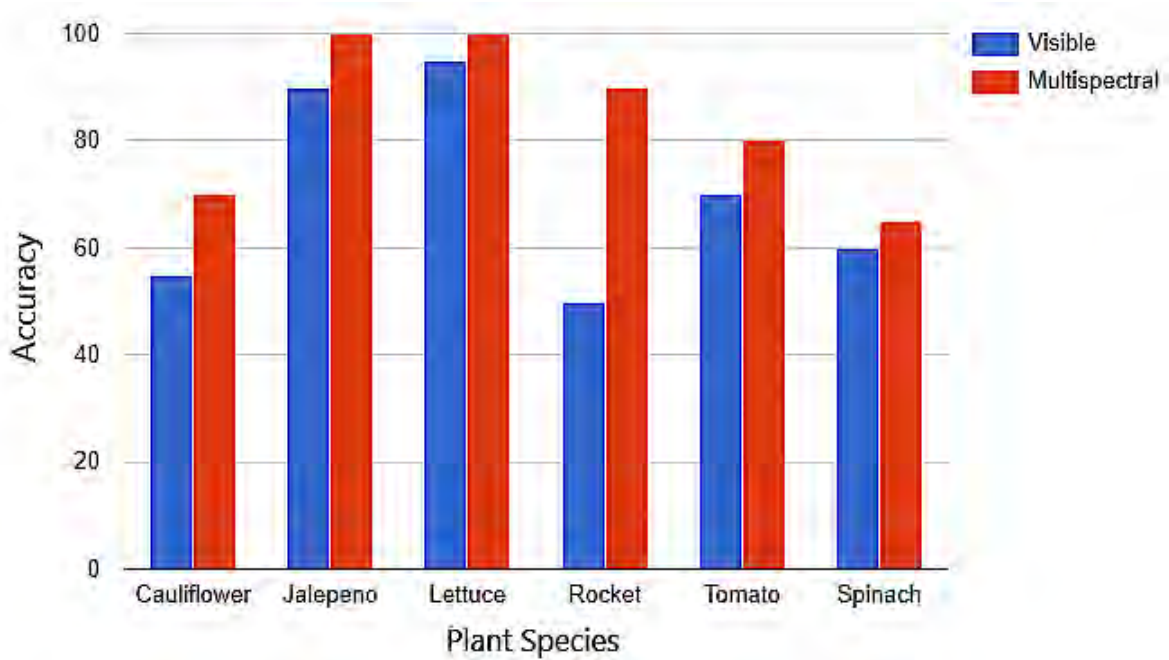


Figure 6.6: Accuracies of multispectral and visible systems classifying healthy and advanced dehydration on separate plant species.

Compared to the results from Experiment 2.1, all classes achieve improved accuracy. As expected, this demonstrates that with *advanced dehydration* where visible leaf symptoms are present, plant stress classification is easier. While multispectral imaging does introduce an impressive overall average increase in accuracy of 14.17%, we hypothesize that it is with the early dehydration classification where the benefits of multispectral input will be most apparent.

6.2.1.3 Experiment 2.3

Experiment 2.3 trains and tests individual binary classification models on *healthy* or *early* stages of dehydration per plant species for visible and multispectral input. Figure 6.7 shows that in the case of *early dehydration* detection, visible input obtains an average accuracy of 60.83% and multispectral input obtains an average accuracy of 89.17%. Multispectral input outperforms visible input by an average of 28.34% with jalapeno and

lettuce obtaining 100% accuracies. This is almost double the increase demonstrated in Experiment 2.2. This significant increase is attributed to the multispectral input's sensitivity to changes of far-red and NIR reflectance caused by stress in the leaf. These changes in reflectance are not as apparent in the visible light wavelengths (Carter and Knapp, 2001, Gitelson *et al.*, 2003, Jones and Vaughan, 2010) and thus are more challenging for the visible models to detect.

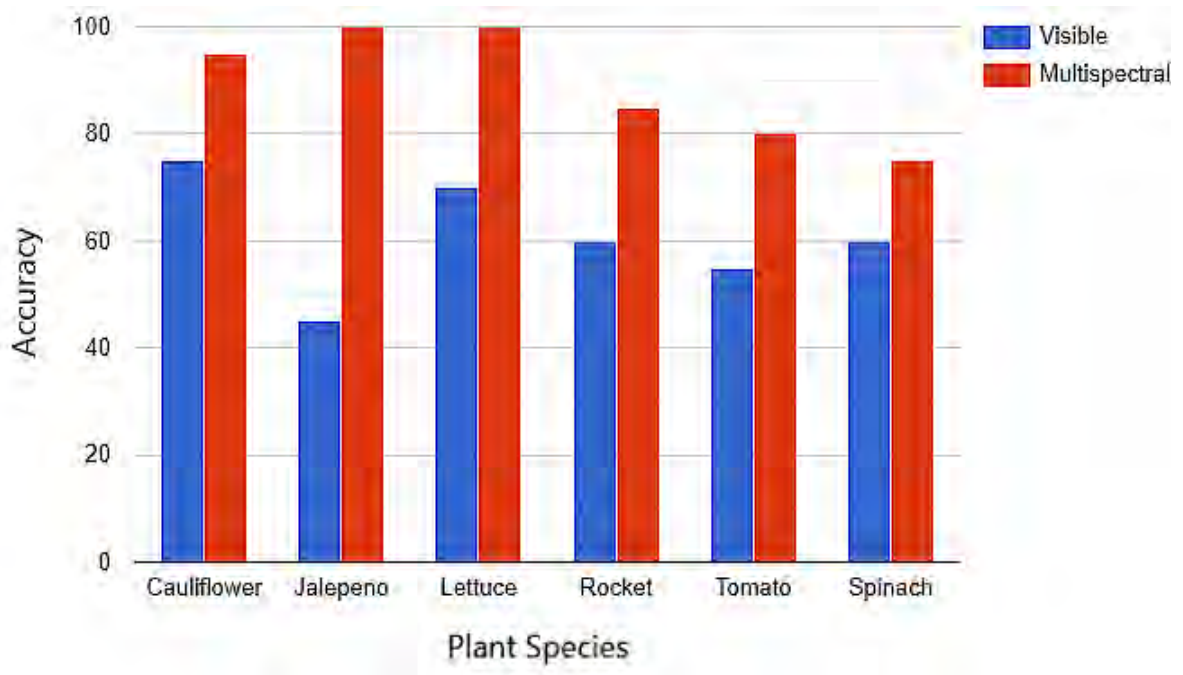


Figure 6.7: Accuracies of multispectral and visible systems classifying healthy and early dehydration on separate plant species.

Furthermore, resuming watering the advanced dehydrated plants showed that the fastest to recover were spinach, rocket, cauliflower, and tomato. This suggests that those plants could be considered more ‘drought resistant’ than lettuce and jalapeno — the majority of which did not recover. In this case, ‘drought resistance’ refers to the quick recovery of a plant when watered after being in a state of dehydration. It does not imply the plant will thrive in drought conditions.

With this information, it can be seen in Figure 6.7 that the less ‘drought resistant’ plants obtain higher classification accuracies. Obtaining high accuracies for plants that struggle to recover from dehydration is of high importance for avoiding crop loss. The models that

obtain the highest accuracies across all experiments were those of jalapeno and lettuce — the plants presented as less likely to recover from dehydration.

6.2.1.4 Summary of Experiment 2

It is evident across all experiments above that multispectral input improves model accuracy for plant dehydration stress detection systems. The models perform better on less ‘drought resistant’ plants — the plants with poor dehydration recovery and more likely to benefit from early dehydration detection to avoid crop loss.

These models were trained on the collected plant species separately and their performances represent the system’s ability to handle water stress detection when the plant species is already known — as is typically the case in agriculture.

6.2.2 All Plant Species

All experiments conducted below train models on the full collected dataset of all plant species.

6.2.2.1 Experiment 3.1

Experiment 3.1 trains and tests multiclass classification models on *healthy*, *early* or *advanced dehydration* stages on all plant species for visible and multispectral input. The data used in this experiment consists of sample images from all plant species divided into three target classes, denoting only the dehydration stage. This produces a single generalized model that can be applied to input of any plant species used during training and thus classify the stage of dehydration stress irrespective of the plant species. This experiment provides the baseline results to which Experiments 3.2 and 3.3 can be compared.

Table 6.5 shows that multispectral input introduces a 12.22% increase in accuracy as compared to visible input. This is consistent with the results obtained in Experiment 2.1

to 2.3. Overall, the multispectral system obtains a 62.22% accuracy which is a 2.23% drop from the overall average accuracy of the individual plant species in Experiment 2.1. Thus, a general plant stress system is viable with a small compromise on accuracy.

Table 6.5: Performance of the multispectral and visible systems classifying healthy, early, and advanced dehydration on all plant species.

Dataset	Training		Test	
	<i>Accuracy</i>	<i>F-score</i>	<i>Accuracy</i>	<i>F-score</i>
Visible	47.90	30.00	50.00	42.29
Multispectral	64.41	63.08	62.22	57.83

6.2.2.2 Experiment 3.2

Experiment 3.2 trains and tests binary classification models on *healthy* or *advanced dehydration* stages on all plant species for visible and multispectral input.

Table 6.6 shows that the visible model obtains a 77.50% accuracy when identifying *advanced dehydration* with visible leaf symptoms. However, multispectral input outperforms visible input by 15.00% with an overall accuracy of 92.50%. This demonstrates that even with visible dehydration symptoms on the leaf, the multispectral input provides extra reflectance information that ResNet-44 is able to utilize to improve the overall classification performance.

Table 6.6: Performance of the multispectral and visible systems classifying healthy and advanced dehydration on all plant species.

Dataset	Training		Test	
	<i>Accuracy</i>	<i>F-score</i>	<i>Accuracy</i>	<i>F-score</i>
Visible	86.71	86.71	77.50	77.50
Multispectral	95.27	95.27	92.50	92.50

Compared with the results of classifying the stage of dehydration in Experiment 3.1, it is clear that identifying advanced dehydration is trivial for a multispectral system with

an increase of over 30% accuracy and overall accuracy of 92.50%. Early water stress detection, however, is assumed to be not as trivial.

6.2.2.3 Experiment 3.3

Experiment 3.3 trains and tests binary classification models on *healthy* or *early dehydration* stages of all plant species for visible and multispectral input.

With the challenging task of *early dehydration* detection, it is expected for model accuracies to decrease due to fewer visible stress indicators present on the leaf. However, as seen in Table 6.7, both models still perform relatively well. Multispectral input outperforms visible input by 9.17% with an overall accuracy of 85.00%. Classifying only early dehydration gives an increase of over 20%, compared with Experiment 3.1 — where all three classes were classified.

Table 6.7: Performance of the multispectral and visible systems classifying healthy and early dehydration on all plant species.

Dataset	Training		Test	
	<i>Accuracy</i>	<i>F-score</i>	<i>Accuracy</i>	<i>F-score</i>
Visible	84.46	84.46	75.83	75.83
Multispectral	92.34	92.34	85.00	85.00

The contingency table of the test output produced by the multispectral and visible systems on 120 test images is given in Table 6.8. The absence of overlap in incorrect classifications indicates that the two systems do not share the same classification weak points.

Table 6.8: Distribution of test output of the multispectral and visible systems classifying early dehydration.

Visible	Multispectral	
	<i>Correct</i>	<i>Incorrect</i>
<i>Correct</i>	81	10
<i>Incorrect</i>	29	0

Conducting a McNemar test on the data from Table 6.8 produces the McNemar statistics of $\chi^2 = 10$ and $p = 0.0034$. Due to the low values in the contingency table, an exact p-value is calculated using a binomial distribution. It is evident that $p < 0.05$, which rejects the null hypothesis and indicates a highly significant difference between the two classification models. Therefore, multispectral input significantly outperforms visible input for early dehydration classification. This supports the research hypothesis.

6.2.2.4 Summary of Experiment 3

Experiment 3 tested the system’s dehydration stress detection performance, irrespective of the plant species the stress was present in. As with Experiment 2, multispectral input consistently outperformed visible input. A McNemar test conducted on the visible and multispectral systems classifying *early dehydration* shows that the difference in performance between the systems is statistically significant. Therefore, for early dehydration, multispectral input significantly improves classification performance. This supports the research hypothesis.

Furthermore, as expected, the simpler binary classifications produced higher accuracies than multiclass classifications during water stress detection. This suggests that a combination of the two binary systems could potentially outperform the three-class model for determining the stage of dehydration. It also suggests that combining both the *early* and *advanced dehydration* classes into one class representing the dehydration stress as a whole could potentially produce a binary classification model with higher stress classification accuracies than the multiclass model.

6.2.2.5 Experiment 4.1

Experiment 4.1 trains and tests the multitask¹ classification models of both *plant species* and *healthy, early or advanced dehydration* stages for visible and multispectral input.

¹A classifier with multiple outputs — species and stress stages.

The complexity of identifying both the plant species and stage of dehydration is evident in the low results obtained in Table 6.9. Multispectral input performed the best with a 39.44% accuracy, outperforming the visible model by 2.77%. The low performance could be due to the small dataset size relative to the complexity of the task.

Table 6.9: Performance of the multispectral and visible systems classifying plant species and healthy, early or advanced dehydration.

Dataset	Training		Test	
	<i>Accuracy</i>	<i>F-score</i>	<i>Accuracy</i>	<i>F-score</i>
Visible	39.94	24.78	36.67	17.56
Multispectral	47.90	37.98	39.44	29.06

As done in the previous experiments, we can split the classification problem into two: *early dehydration* detection and *advanced dehydration* detection. This will either improve classification accuracy or highlight which stage of dehydration is causing confusion.

6.2.2.6 Experiment 4.2

Experiment 4.2 trains and tests the multitask classification models of both *plant species* and *healthy* or *advanced dehydration* stages for visible and multispectral input.

The results, given in Table 6.10, emphasize the benefit of using multispectral input over visible input, with multispectral input obtaining an accuracy over 20% higher than that of visible input. It is interesting to note that their difference in F-score is even higher, indicating a greater overall performance by the multispectral input.

Table 6.10: Performance of the multispectral and visible systems classifying plant species and healthy or advanced dehydration.

Dataset	Training		Test	
	<i>Accuracy</i>	<i>F-score</i>	<i>Accuracy</i>	<i>F-score</i>
Visible	55.86	42.88	48.33	38.60
Multispectral	94.14	93.85	71.67	72.17

6.2.2.7 Experiment 4.3

Experiment 4.3 trains and tests the multitask classification models of both *plant species* and *healthy* or *early dehydration* stages for visible and multispectral input.

Compared to the accuracies produced by the model classifying stage of dehydration in Experiment 4.1, the results given in Table 6.11 show an average improvement of only 3.61%. This suggests that the confusion in Experiment 4.1 was potentially introduced by the *early dehydration* class.

Table 6.11: Performance of the multispectral and visible systems classifying plant species and healthy or early dehydration.

Dataset	Training		Test	
	<i>Accuracy</i>	<i>F-score</i>	<i>Accuracy</i>	<i>F-score</i>
Visible	39.19	24.53	40.83	27.03
Multispectral	54.95	41.21	42.50	29.81

Inspection of the confusion matrix produced by the multispectral model and given in Figure 6.8 shows that the majority of the confusion that exists is between overlapping classes. This is where either the stage of dehydration was correct, but plant species was not, or the plant species was correct, but stage of dehydration was not. This is highlighted by the high confusion of *Tomato-Healthy* with *Tomato-Early Dehydration*, *Spinach-Healthy* with *Spinach-Early Dehydration*, *Lettuce-Early Dehydration* with *Jalepeno-Early Dehydration*, *Rocket-Healthy* with *Cauliflower-Healthy* and *Lettuce-Healthy*, etc.

This suggests that the system performs well when generalizing plant species or generalizing stage of dehydration individually, but not both simultaneously.

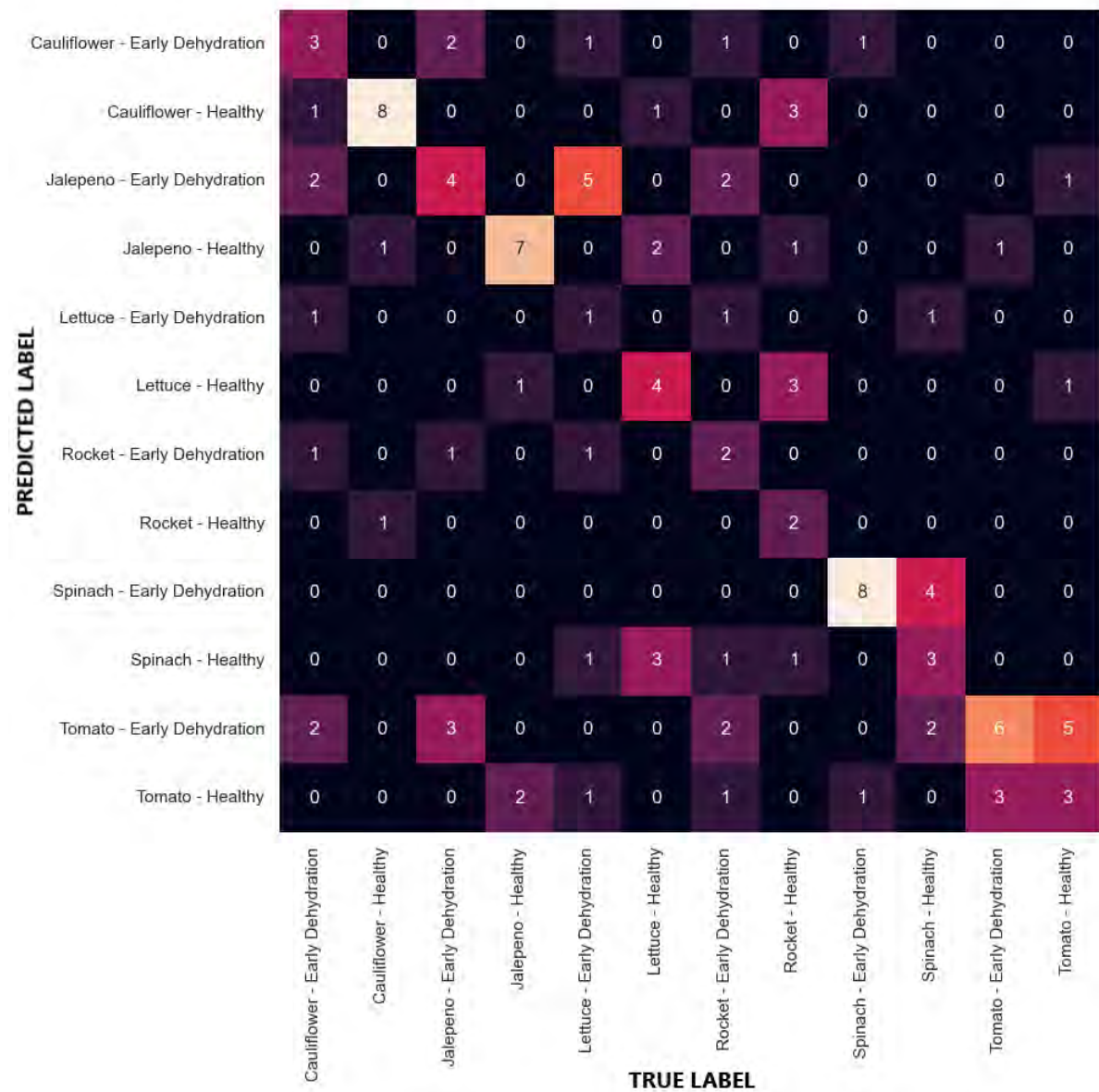


Figure 6.8: Confusion matrix produced by the multispectral model in Experiment 4.3. Lighter colors indicate higher concentrations of test predictions with correct predictions found along the diagonal.

6.2.2.8 Summary of Experiment 4

While multispectral input did show an improvement to all models tested, it is evident that the system overall does not perform well when classifying both the stage of dehydration and its subclass, i.e. plant species. This is not a crucial issue since, as previously mentioned, in agriculture it is typically already known what plant species is being grown and therefore being monitored. However, it also suggests that instead two phases of classification can be performed — plant species first, followed by stage of dehydration on the identified species.

6.3 Summary

This chapter presented and discussed the results obtained from all the experiments conducted as well as validated any parameters used in the implementation of the system.

It was found that out of those considered, ResNet-44 was the top-performing architecture for plant disease and stress classification with both visible and multispectral input and, therefore, was used for all models generated henceforth.

Throughout all tests conducted, the results emphasize the clear improvement to model performance that multispectral input provides. This was first validated on a collected plant disease dataset in preliminary experiments.

In a larger set of experiments, the multispectral system performed best when detecting stage of stress on known plant species. This is useful when the plant species are already known, as is typically the case in agriculture. It required separate models to be trained for each plant species and each stage of stress. The multispectral input gave an average improvement of 14.17% to advanced dehydration detection and 28.33% to early dehydration detection.

A similar trend was observed on the visible and multispectral systems identifying the stage of dehydration on all plant species. Applying McNemar's test showed a statistically

significant difference in performance between multispectral and visible *early dehydration* detection. This demonstrated the multispectral input's ability to better detect the minor changes of leaf reflectance in the far-red and NIR wavelengths during the initial stages of stress. This also supports the research hypothesis.

The system was found to have strong discrimination of stress stage and species separately; however, it did not multitask well to classify them simultaneously. This would suggest using multiple models would provide the best performance by the system. Overall it was speculated that the model performance for all models tested would improve with the use of a larger training dataset.

7

Conclusion and Future Work

This chapter concludes the thesis, highlights the contributions made towards the research, and provides directions for future work.

7.1 Conclusion

In this research, machine learning algorithms combined with multispectral input were investigated to improve early plant disease and stress detection. The system utilized input from the far-red and NIR wavelength range. This was proposed by the related literature as a wavelength range more sensitive to the changes in reflectance caused by stress in a leaf than the visible wavelength range.

To provide an accurate comparison between systems using visible versus multispectral input, an appropriate dataset was required. Two datasets were thus collected: a small disease dataset and a larger early and advanced dehydration stress dataset. The small disease dataset was used to evaluate the model architectures for both visible and multispectral input and indicate the system's applicability to disease data. The early and advanced dehydration stress dataset was used to test the system's performance in general stress and early stress detection and to evaluate the effect of multispectral input over visible input in both cases.

For the machine learning algorithm, three CNN architectures were evaluated. ResNet-44 was found to be a consistent top performer on plant disease and stress data and was therefore used for all further experimentation. The final system was constructed using the following steps: model building, tuning, training, and then testing. The dataset was divided into the following subsections: training, validation, and testing data with data augmentation applied to the training data only. The training and validation data were utilized during model building, tuning, and training. The unseen test data was used for model testing. This resulted in models explicitly optimized for our collected dataset.

Overall, nine tests were conducted on the early and advanced dehydration dataset. Throughout all tests, the multispectral systems consistently outperformed the visible systems. The multispectral system performed best for the detection of stages of stress on known plant species. This is useful when the plant species are already known, as is typically the case in agriculture. It required separate models to be trained for each plant species and for each stage of stress. Compared with the visible system, the multispectral system showed an average improvement of 14.17% on advanced dehydration detection and 28.33% on early dehydration detection. McNemar's test showed a statistically significant difference in performance between multispectral and visible *early dehydration* detection. This demonstrated the multispectral system's ability to better detect the minor changes of leaf reflectance in the far-red and NIR wavelengths during the initial stages of stress. This, therefore, supports the hypothesis that a system can be constructed to identify early plant stress symptoms with improved accuracy by leveraging multispectral imaging instead of visible imaging.

To conclude, the overall findings showed that the system effectively classified plant species and stages of dehydration separately, however, classifying both simultaneously was ineffective. This suggests that using multiple output models can provide better performance. The accuracy for all models tested would likely improve with the use of a larger training dataset. Furthermore, multispectral imaging provides a promising avenue for future early stress detection research.

7.2 Contributions

All seven objectives were successfully achieved during this study:

1. Visible and multispectral image datasets were collected in a real-world complex background setting.
2. A relatively inexpensive system to perform multitask classification of both plant species and stage of plant stress was designed and created.
3. The performance of state-of-the-art CNN architectures trained on plant disease image datasets were compared.
4. The relative effectiveness of visible and multispectral input for classifying *healthy* versus *advanced* stress on multiple plant species was established.
5. The relative effectiveness of visible and multispectral input for classifying *healthy* versus *early* stress on multiple plant species was established.
6. The relative effectiveness of visible and multispectral input for classifying *healthy*, *early* and *advanced* stress on multiple plant species was established.
7. The relative effectiveness of visible and multispectral input for multitask classification *plant species* as well as state of *healthy*, *early* or *advanced* stress on multiple plant species was established.

The completed objectives yielded three significant contributions to the body of knowledge:

1. Publicly available proof-of-concept multispectral and visible spinach datasets containing *healthy* and *diseased* plants.
2. Publicly available multispectral and visible datasets containing *healthy*, *early* or *advanced* dehydrated plants.
3. A results contribution including the evaluation of three CNN-based architectures on popular plant disease datasets and aforementioned collected datasets.

7.3 Future Work

Possible future work includes the following additional objectives:

1. Collect appropriate datasets and, thus, create systems for other plant stresses such as disease and pest infestation.
2. Expand the collected plant dehydration dataset to provide a large training sample size and, therefore, produce a higher system accuracy.
3. Effectively combine appropriate models to produce one top-performing system for plant species and stress classification instead of the single low performing multiclass classification model produced.
4. Create a system to predict the measurement of the plant's stress, i.e. leaf water content for dehydration, and thus the stage of plant stress instead of classifying the input directly into class bins. This would require a more detailed dataset to be collected.
5. Develop a mobile application for the system to be used in the field with easily available multispectral input devices such as mobile camera adapters.

References

- Albawi, S., Mohammed, T. A., and Al-Zawi, S.** Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6. IEEE, 2017.
- Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Hasan, M., Van Essen, B. C., Awwal, A. A., and Asari, V. K.** A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 8(3):292, 2019.
- Amthor, J. S.** From sunlight to phytomass: on the potential efficiency of converting solar radiation to phyto-energy. *New Phytologist*, 188(4):939–959, 2010.
- Anderson, J. A.** An introduction to neural networks. MIT press, 1995.
- Ausmus, B. S. and Hilty, J. W.** Reflectance studies of healthy, maize dwarf mosaic virus-infected, and helminthosporium maydis-infected corn leaves. *Remote Sensing of Environment*, 2:77–81, 1971.
- Bengio, Y., Simard, P., and Frasconi, P.** Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.
- Bergstra, J. and Bengio, Y.** Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.
- Blackburn, G. A. and Ferwerda, J. G.** Retrieval of chlorophyll concentration from leaf reflectance spectra using wavelet analysis. *Remote Sensing of Environment*, 112(4):1614–1632, 2008.

- Bock, C., Poole, G., Parker, P., and Gottwald, T.** Plant disease severity estimated visually, by digital photography and image analysis, and by hyperspectral imaging. *Critical Reviews in Plant Sciences*, 29(2):59–107, 2010.
- Bravo, C., Moshou, D., West, J., McCartney, A., and Ramon, H.** Early disease detection in wheat fields using spectral reflectance. *Biosystems Engineering*, 84(2):137–145, 2003.
- Carter, G. A. and Knapp, A. K.** Leaf optical properties in higher plants: linking spectral characteristics to stress and chlorophyll concentration. *American journal of botany*, 88(4):677–684, 2001.
- Cherkassky, V. and Mulier, F. M.** Learning from data: concepts, theory, and methods. John Wiley & Sons, 2007.
- Chouhan, S. S., Singh, U. P., Kaul, A., and Jain, S.** A data repository of leaf images: Practice towards plant conservation with plant pathology. In *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*, pages 700–707. IEEE, 2019.
- Cunningham, P., Cord, M., and Delany, S. J.** Supervised learning. In *Machine learning techniques for multimedia*, pages 21–49. Springer, 2008.
- Dean, R. A., Talbot, N. J., Ebbole, D. J., Farman, M. L., Mitchell, T. K., Orbach, M. J., Thon, M., Kulkarni, R., Xu, J.-R., Pan, H. et al.** The genome sequence of the rice blast fungus *magnaporthe grisea*. *Nature*, 434(7036):980–986, 2005.
- Dietterich, T.** Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, 27(3):326–327, 1995.
- Domingos, P.** A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.
- Falkner, S., Klein, A., and Hutter, F.** Combining hyperband and bayesian optimization. In *NIPS 2017 Bayesian Optimization Workshop (Dec 2017)*. 2017.

- Ferentinos, K. P.** Deep learning models for plant disease detection and diagnosis. *Computers and Electronics in Agriculture*, 145:311–318, 2018.
- Gitelson, A. A., Gritz, Y., and Merzlyak, M. N.** Relationships between leaf chlorophyll content and spectral reflectance and algorithms for non-destructive chlorophyll assessment in higher plant leaves. *Journal of plant physiology*, 160(3):271–282, 2003.
- Guo, T., Dong, J., Li, H., and Gao, Y.** Simple convolutional neural network on image classification. In *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, pages 721–724. IEEE, 2017.
- Halevy, A., Norvig, P., and Pereira, F.** The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.
- Harish, B., Hedge, A., Venkatesh, O., Spoorthy, D., and Sushma, D.** Classification of plant leaves using morphological features and zernike moments. In *2013 international conference on advances in computing, communications and informatics (ICACCI)*, pages 1827–1831. IEEE, 2013.
- He, K., Zhang, X., Ren, S., and Sun, J.** Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. 2016.
- Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R.** Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.
- Huang, C.-M., Lee, Y.-J., Lin, D. K., and Huang, S.-Y.** Model selection for support vector machines via uniform design. *Computational Statistics & Data Analysis*, 52(1):335–346, 2007.
- Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q.** Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708. 2017.

- Hughes, D. P. and Salathé, M.** An open access repository of images on plant health to enable the development of mobile disease diagnostics through machine learning and crowdsourcing. *CoRR*, abs/1511.08060, 2015.
URL <http://arxiv.org/abs/1511.08060>
- Hussain, M., Bird, J. J., and Faria, D. R.** A study on cnn transfer learning for image classification. In *UK Workshop on computational Intelligence*, pages 191–202. Springer, 2018.
- Ioffe, S. and Szegedy, C.** Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.
- Jamieson, K. and Talwalkar, A.** Non-stochastic best arm identification and hyperparameter optimization. In *Artificial Intelligence and Statistics*, pages 240–248. PMLR, 2016.
- Janwale, A.** Plant leaves image segmentation techniques: A review. *International Journal of Computer Sciences and Engineering*, 5:147–150, 06 2017.
- Jiménez, Á. B., Lázaro, J. L., and Dorronsoro, J. R.** Finding optimal model parameters by deterministic and annealed focused grid search. *Neurocomputing*, 72(13-15):2824–2832, 2009.
- Jones, D. R.** A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21(4):345–383, 2001.
- Jones, H. G. and Vaughan, R. A.** Remote sensing of vegetation: principles, techniques, and applications. Oxford university press, 2010.
- Khaled, A. Y., Abd Aziz, S., Bejo, S. K., Nawi, N. M., Seman, I. A., and Onwude, D. I.** Early detection of diseases in plant tissue using spectroscopy—applications and limitations. *Applied Spectroscopy Reviews*, 53(1):36–64, 2018.

- Khan, A., Sohail, A., Zahoor, U., and Qureshi, A. S.** A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 53(8):5455–5516, 2020.
- Kim, Y., Glenn, D. M., Park, J., Ngugi, H. K., and Lehman, B. L.** Hyperspectral image analysis for plant stress detection. In *2010 Pittsburgh, Pennsylvania, June 20–June 23, 2010*, page 1. American Society of Agricultural and Biological Engineers, 2010.
- Krizhevsky, A., Hinton, G. et al.** Learning multiple layers of features from tiny images. *Master’s thesis, University of Toronto*, 2009.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E.** Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.
- Labach, A., Salehinejad, H., and Valaee, S.** Survey of dropout methods for deep neural networks. *arXiv preprint arXiv:1904.13310*, 2019.
- LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P.** Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- Li, L., Jamieson, K. G., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A.** Efficient hyperparameter optimization and infinitely many armed bandits. *CoRR*, abs/1603.06560, 16, 2016.
- Loey, M., ElSawy, A., and Afify, M.** Deep learning in plant diseases detection for agricultural crops: a survey. *International Journal of Service Science, Management, Engineering, and Technology (IJSSMET)*, 11(2):41–58, 2020.
- López, M. M., Bertolini, E., Olmos, A., Caruso, P., Gorris, M. T., Llop, P., Penyalver, R., and Cambra, M.** Innovative tools for detection of plant pathogenic viruses and bacteria. *International Microbiology*, 6(4):233–243, 2003.
- Lu, J., Tan, L., and Jiang, H.** Review on convolutional neural network (cnn) applied to plant leaf disease classification. *Agriculture*, 11(8):707, 2021.

- Malthus, T. J. and Madeira, A. C.** High resolution spectroradiometry: spectral reflectance of field bean leaves infected by botrytis fabae. *Remote Sensing of Environment*, 45(1):107–116, 1993.
- Martinelli, F., Scalenghe, R., Davino, S., Panno, S., Scuderi, G., Ruisi, P., Villa, P., Stroppiana, D., Boschetti, M., Goulart, L. R. et al.** Advanced methods of plant disease detection. a review. *Agronomy for Sustainable Development*, 35(1):1–25, 2015.
- McNemar, Q.** Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947.
- Mockus, J., Tiesis, V., and Zilinskas, A.** The application of bayesian methods for seeking the extremum. *Towards global optimization*, 2(117-129):2, 1978.
- Mohanty, S. P., Hughes, D. P., and Salathé, M.** Using deep learning for image-based plant disease detection. *Frontiers in plant science*, 7:1419, 2016.
- Navulur, K.** Multispectral image analysis using the object-oriented paradigm. CRC press, 2006.
- Nielsen, M. A.** Neural networks and deep learning, volume 25. Determination press San Francisco, CA, 2015.
- Nouri, M., Gorretta, N., Vaysse, P., Giraud, M., Germain, C., Keresztes, B., and Roger, J.-M.** Near infrared hyperspectral dataset of healthy and infected apple tree leaves images for the early detection of apple scab disease. *Data in brief*, 16:967–971, 2018.
- O’Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., Invernizzi, L. et al.** Keras tuner. <https://github.com/keras-team/keras-tuner>, 2019.
- Padol, P. B. and Yadav, A. A.** Svm classifier based grape leaf disease detection. In *2016 Conference on advances in signal processing (CASP)*, pages 175–179. IEEE, 2016.
- Pavia, D. L., Lampman, G. M., Kriz, G. S., and Vyvyan, J. A.** Introduction to spectroscopy. Cengage learning, 2014.

- Peñuelas, J., Filella, I., Biel, C., Serrano, L., and Save, R.** The reflectance at the 950–970 nm region as an indicator of plant water status. *International journal of remote sensing*, 14(10):1887–1905, 1993.
- Pitts, D. G. and Kleinstein, R.** The electromagnetic spectrum. *Environmental Vision*. Boston, Butterworth-Heinemann, pages 87–135, 1993.
- Prechelt, L.** Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761–767, 1998.
- Priya, C. A., Balasaravanan, T., and Thanamani, A. S.** An efficient leaf recognition algorithm for plant classification using support vector machine. In *International conference on pattern recognition, informatics and medical engineering (PRIME-2012)*, pages 428–432. IEEE, 2012.
- Ranzato, M., Huang, F. J., Boureau, Y.-L., and LeCun, Y.** Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007.
- Raskutti, G., Wainwright, M. J., and Yu, B.** Early stopping and non-parametric regression: an optimal data-dependent stopping rule. *The Journal of Machine Learning Research*, 15(1):335–366, 2014.
- Rawat, W. and Wang, Z.** Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017.
- Riedell, W. E. and Blackmer, T. M.** Leaf reflectance spectra of cereal aphid-damaged wheat. *Crop Science*, 39(6):1835–1840, 1999.
- Ripley, B. D.** Pattern recognition and neural networks. Cambridge university press, 2007.
- Rogachev, A. F. and Melikhova, E. V.** Automation of the process of selecting hyper-parameters for artificial neural networks for processing retrospective text information.

- IOP Conference Series: Earth and Environmental Science*, 577:012012, oct 2020. doi: 10.1088/1755-1315/577/1/012012.
URL <https://doi.org/10.1088/1755-1315/577/1/012012>
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. et al.** Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.
- Schaare, P. and Fraser, D.** Comparison of reflectance, interactance and transmission modes of visible-near infrared spectroscopy for measuring internal properties of kiwifruit (*actinidia chinensis*). *Postharvest Biology and Technology*, 20(2):175–184, 2000.
- Shimodaira, H.** Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.
- Shorten, C. and Khoshgoftaar, T. M.** A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.
- Shruthi, U., Nagaveni, V., and Raghavendra, B.** A review on machine learning classification techniques for plant disease detection. In *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, pages 281–284. IEEE, 2019.
- Simard, P. Y., Steinkraus, D., Platt, J. C. et al.** Best practices for convolutional neural networks applied to visual document analysis. In *Icdar*, volume 3. 2003.
- Simonyan, K. and Zisserman, A.** Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- Sims, D. A. and Gamon, J. A.** Relationships between leaf pigment content and spectral reflectance across a wide range of species, leaf structures and developmental stages. *Remote sensing of environment*, 81(2-3):337–354, 2002.

- Singh, K., Gupta, I., and Gupta, S.** Svm-bdt pnn and fourier moment technique for classification of leaf shape. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 3(4):67–78, 2010.
- Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., and Stefanovic, D.** Deep neural networks based recognition of plant diseases by leaf image classification. *Computational intelligence and neuroscience*, 2016, 2016.
- Snoek, J., Larochelle, H., and Adams, R. P.** Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R.** Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.
- Su, W.-H. and Sun, D.-W.** Multispectral imaging for plant food quality analysis and visualization. *Comprehensive reviews in food science and food safety*, 17(1):220–239, 2018.
- Sun, D.-W.** Infrared spectroscopy for food quality analysis and control. Academic press, 2009.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A.** Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9. 2015.
- Tan, H. H. and Lim, K. H.** Vanishing gradient mitigation with deep learning neural network optimization. In *2019 7th international conference on smart computing & communications (ICSCC)*, pages 1–4. IEEE, 2019.
- Tan, M. and Le, Q.** Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

- Thornton, C., Hutter, F., Hoos, H. H., and Leyton-Brown, K.** Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855. 2013.
- Verhoeven, G.** The reflection of two fields: electromagnetic radiation and its role in (aerial) imaging. *AARGNews*, 55:13–18, 2017.
- Walleign, S., Polceanu, M., and Buche, C.** Soybean plant disease identification using convolutional neural network. In *The thirty-first international flairs conference*. 2018.
- Wiesler, S. and Ney, H.** A convergence analysis of log-linear training. *Advances in Neural Information Processing Systems*, 24:657–665, 2011.
- Xu, H., Ying, Y., Fu, X., and Zhu, S.** Near-infrared spectroscopy in detecting leaf miner damage on tomato leaf. *Biosystems Engineering*, 96(4):447–454, 2007.
- Yang, W., Chen, J., Chen, G., Wang, S., and Fu, F.** The early diagnosis and fast detection of blast fungus, *magnaporthe grisea*, in rice plant by using its chitinase as biochemical marker and a rice cdna encoding mannose-binding lectin as recognition probe. *Biosensors and Bioelectronics*, 41:820–826, 2013.
- Yu, H., Chen, C., Du, X., Li, Y., Rashwan, A., Hou, L., Jin, P., Yang, F., Liu, F., Kim, J., and Li, J.** TensorFlow Model Garden. <https://github.com/tensorflow/models>, 2020.
- Zeiler, M. D. and Fergus, R.** Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.
- Zovko, M., Žibrat, U., Knapič, M., Kovačić, M. B., and Romić, D.** Hyperspectral remote sensing of grapevine drought stress. *Precision agriculture*, 20(2):335–347, 2019.
- Zygielbaum, A. I., Gitelson, A. A., Arkebauer, T. J., and Rundquist, D. C.** Non-destructive detection of water stress and estimation of relative water content in maize. *Geophysical research letters*, 36(12), 2009.



Additional Results

A.1 Experiment 2.1

Table A.1: Performance of the models classifying healthy, early dehydration and advanced dehydration on separate plant species.

Plant Species	Dataset	Training		Testing	
		<i>Accuracy</i>	<i>F-score</i>	<i>Accuracy</i>	<i>F-score</i>
Cauliflower	Visible	34.23	9.79	40.00	30.77
	Infrared	75.68	75.83	70.00	64.29
Jalepeno	Visible	25.23	6.94	40.00	34.62
	Infrared	76.58	76.58	86.67	88.14
Lettuce	Visible	34.23	14.50	36.67	30.19
	Infrared	95.50	95.02	83.33	83.33
Rocket	Visible	54.05	43.90	40.00	37.74
	Infrared	74.77	70.30	56.67	50.00
Tomato	Visible	32.43	6.11	36.67	20.83
	Infrared	35.14	14.60	40.00	33.33
Spinach	Visible	33.03	10.45	23.33	19.61
	Infrared	53.21	42.35	50.00	46.43

A.2 Experiment 2.2

Table A.2: Performance of the binary models classifying healthy and advanced dehydration on separate plant species.

Plant Species	Dataset	Training		Testing	
		<i>Accuracy</i>	<i>F-score</i>	<i>Accuracy</i>	<i>F-score</i>
Cauliflower	Visible	44.59	44.59	55.00	55.00
	Infrared	95.95	95.95	70.00	70.00
Jalepeno	Visible	100.00	100.00	90.00	90.00
	Infrared	98.65	98.65	100.00	100.00
Lettuce	Visible	97.30	97.30	95.00	95.00
	Infrared	89.19	89.19	100.00	100.00
Rocket	Visible	59.46	59.46	50.00	50.00
	Infrared	78.38	78.38	90.00	90.00
Tomato	Visible	78.38	78.38	70.00	70.00
	Infrared	95.95	95.95	80.00	80.00
Spinach	Visible	44.59	44.59	60.00	60.00
	Infrared	94.59	94.59	65.00	65.00

A.3 Experiment 2.3

Table A.3: Performance of the binary models classifying healthy and early dehydration on separate plant species.

Plant Species	Dataset	Training		Testing	
		<i>Accuracy</i>	<i>F-score</i>	<i>Accuracy</i>	<i>F-score</i>
Cauliflower	Visible	62.16	62.16	75.00	75.00
	Infrared	90.54	90.54	95.00	95.00
Jalepeno	Visible	48.65	48.65	45.00	45.00
	Infrared	100.00	100.00	100.00	100.00
Lettuce	Visible	95.95	95.95	70.00	70.00
	Infrared	98.65	98.65	100.00	100.00
Rocket	Visible	72.97	72.97	60.00	60.00
	Infrared	97.30	97.30	85.00	85.00
Tomato	Visible	50.00	50.00	55.00	55.00
	Infrared	95.95	95.95	80.00	80.00
Spinach	Visible	73.61	73.61	60.00	60.00
	Infrared	86.11	86.11	75.00	75.00

B

Datasets

The complete code and limited data are available at the [Github Repository](#)¹. An alternative location for the data is: [Google Drive](#)². The complete set of data is on available on request via: louisecarmenpoole@gmail.com or d.brown@ru.ac.za email addresses.

The rest of this appendix contains samples of the data used for experiments.

B.1 Spinach Disease Dataset



Figure B.1: Samples from each class of the spinach diseased visible and infrared collected datasets.

¹<https://github.com/louise-poole/A-Multispectral-and-Machine-Learning-Approach-to-Early-Stress-Classification-in-Plants>

²<https://drive.google.com/drive/folders/19xkPoHYBESK-qClnvNg6SR33lSt35k6t?usp=sharing>

B.2 Early and Advanced Dehydration Dataset

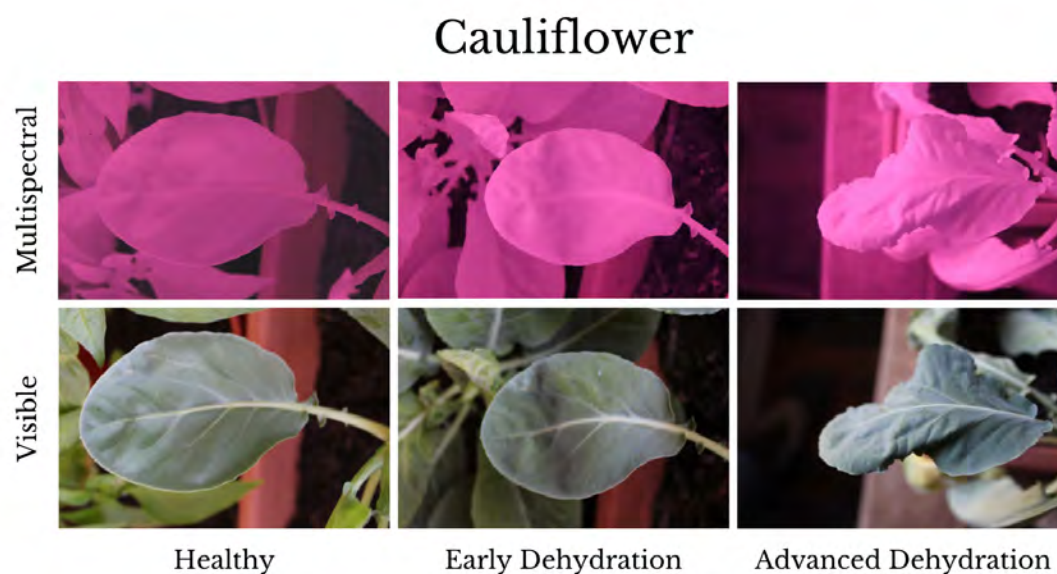


Figure B.2: Samples from each class of the cauliflower visible and infrared collected datasets.



Figure B.3: Samples from each class of the jalapeno visible and infrared collected datasets.

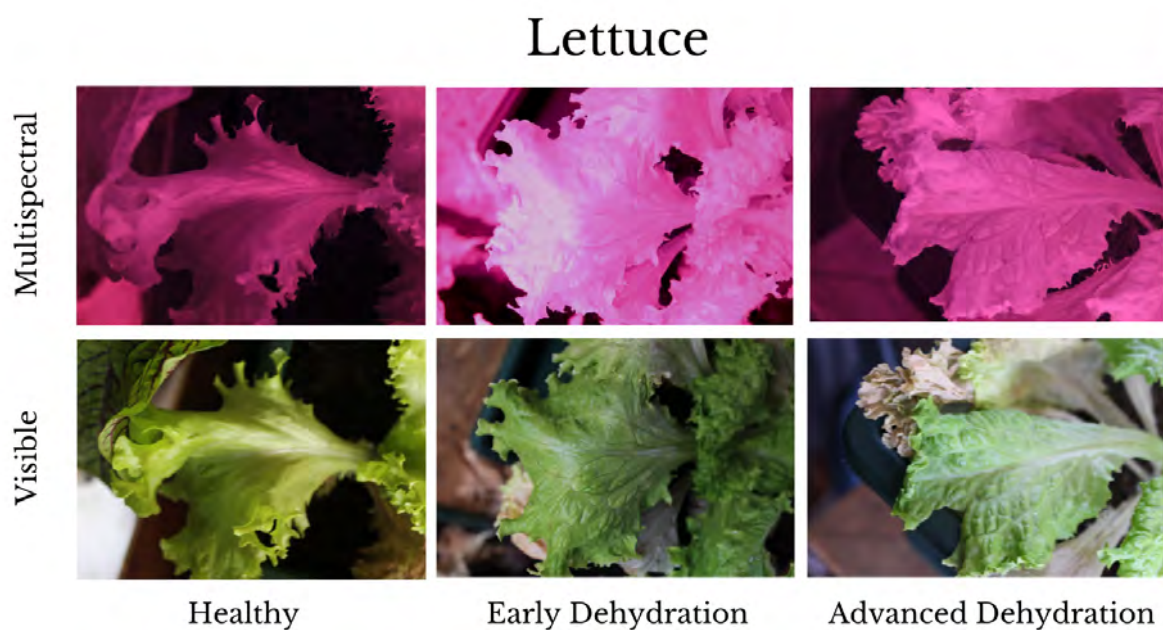


Figure B.4: Samples from each class of the lettuce visible and infrared collected datasets.

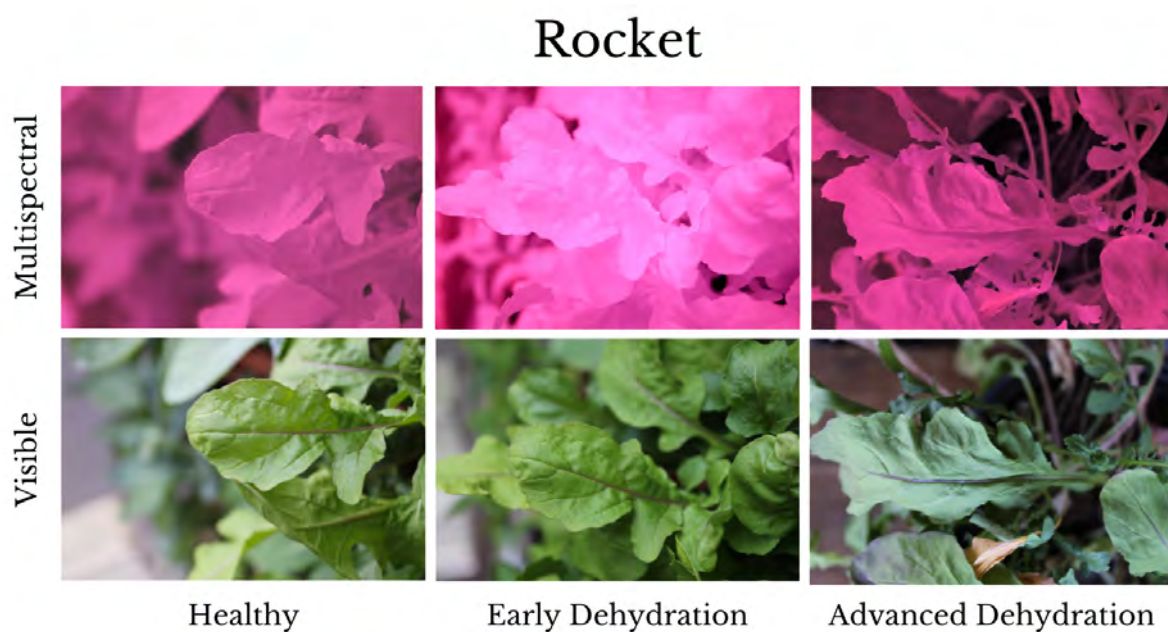


Figure B.5: Samples from each class of the rocket visible and infrared collected datasets.

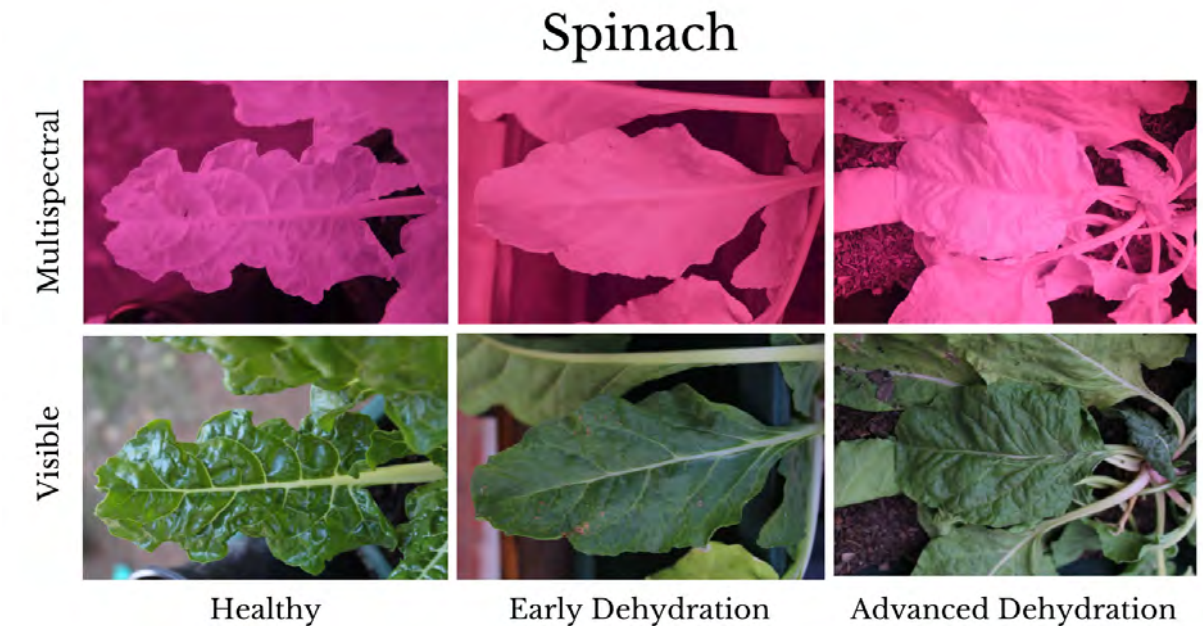


Figure B.6: Samples from each class of the spinach visible and infrared collected datasets.

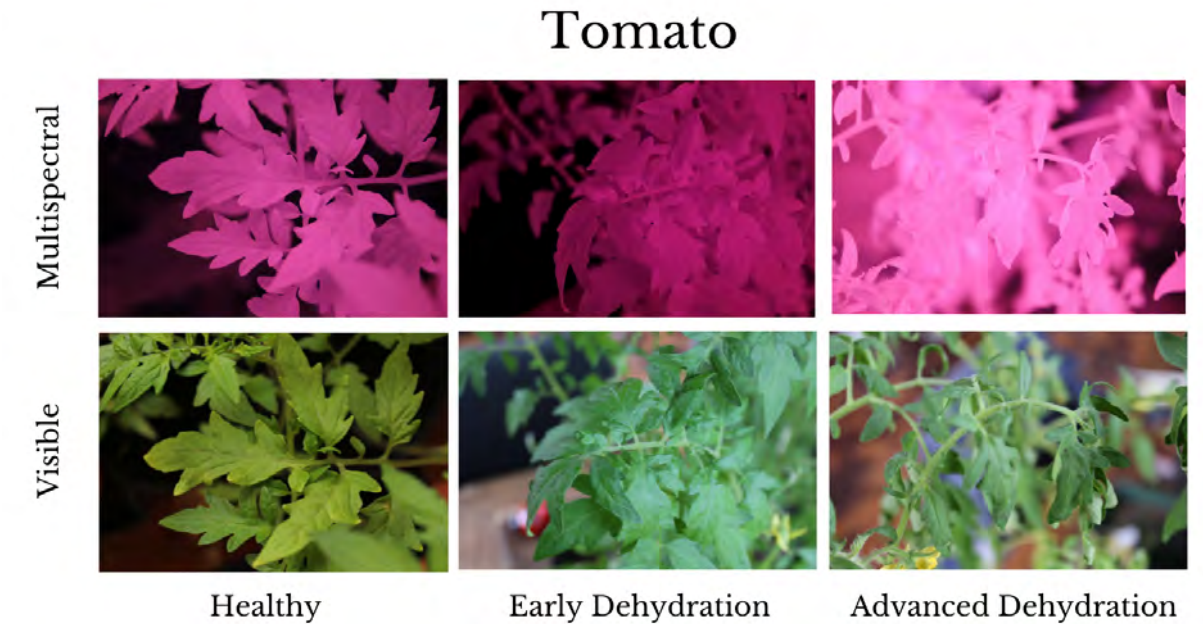


Figure B.7: Samples from each class of the tomato visible and infrared collected datasets.