DEVELOPMENT OF GINGER PLANT HEALTH MONITORING SYSTEM

BY

BU JIA WEI

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

February 2025

COPYRIGHT STATEMENT

This Final Year Project proposal is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project proposal represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project proposal may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I want to extend my sincerest gratitude to my supervisor, Dr. Ng Hui Fuang, for his vast contribution of guidance, encouragement, and support throughout this project. Your expertise and insight have elevated my knowledge and ability to an entirely new level. I am most thankful to have been allowed to work under your charge. This project has been a yardstick in my academic journey and has encouraged me to move beyond excellence in this area of intellectual pursuit.

Specifically, and not least, thanks go to my colleague, Loh Kin Ming, for his unshakeable commitment, intelligence, and camaraderie, which have contributed a great deal of value to this research project. Your leadership in creating the dataset enabled this research, and I am grateful for the working collaboration that came out of it.

I would like to thank Ernest Tan Cong Ying and Wong Kenji too for their generous efforts in annotating the dataset deployed in this project. Your time and effort were extremely crucial in the success of the model training and testing process.

Lastly, I would also wish to offer my sincerest gratitude to my parents and family for their unconditional love, encouragement, support, and belief in me. Your encouragement has been the driving force of my education, and I owe a lifelong debt to the sacrifices you have made to allow me to reach so far.

Without the combined guidance, advice, and encouragement of these fantastic people, the completion of this project would not have been possible. I consider myself extremely lucky and grateful to each one of you. Thank you, all of you.

ABSTRACT

The general intelligent agriculture system is integrated with several other technologies such as sensors, automated irrigation, fertilization, and surveillance systems for increasing efficiency and improving productivity. This project discusses anomaly detection (AD) in images taken from plantations, an important part of such systems. In the related task of plant disease detection, much previous work has relied on various supervised learning approaches; the use of convolutional neural networks and other deep learning models trained on large, annotated datasets of diseased plants has become common. However, this would remain a less feasible approach, considering some practical challenges to acquiring such datasets, particularly in real-world farming scenarios. For example, it is highly impractical and tedious to expect the farmers themselves to take clear, labelled images of every plant. Besides, data collection usually includes flying drones or moving cameras, adding more problems in capturing regular and quality images.

This project applies the unsupervised anomaly detection concept, which avoids the use of large-scale pre-labeled datasets. The proposed system trains the model only on healthy ginger plant images to learn normal patterns and detect deviations, if any, as potential anomalies. Such deviations can be because of a disease in the plant or other health problems. This will not only reduce the overhead of manual data labelling but also enhance the practicality of deploying the system in dynamic agricultural environments. The flagged anomalous images can be used later to augment the supervised learning models, thus enabling hybrid supervision for further refinement of the system.

The strength of the AD model for real-world images of varied natural environmental conditions considers changes in background, light, and climate. Automating plant health monitoring, reduces manual inspections to a minimum, hence allowing farmers to identify health problems much earlier, take remedial measures, and focus on strategic features of crop management. Overall efficiency increases, labour costs are reduced, and much healthier plantations ensue, hence promoting sustainable agriculture. It signals a very promising step toward the exploitation of artificial intelligence to address challenges in modern farming.

Area of Study (Maximum 2): Artificial Intelligence, Computer Vision

Keywords (Maximum 5): Plant Health Monitoring, Anomaly Detection, Unsupervised

Learning, Image Processing, Smart Agriculture

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

OCC	One-Class Classification
ROI	Region of Interest
IPP	Integrated Performance Primitives
CNN	Convolutional Neural Network
UI	User Interface
AD	Anomaly Detection
VLLM	Vision Large Language Model
VLM	Vision-Language Model
OOP	Object-Oriented Programming
API	Application Programming Interface
ABOD	Angle-base Outlier Detection
CLUSTER	Clustering-Based Local Outlier
COF	Connectivity-Based Outlier Factor
HISTOGRAM	Histogram-Based Outlier Detection
IFOREST	Isolation Forest
k-NN	k-Nearest Neighbors Destector
LOF	Local Outlier Factor
OCSVM	One-class Support Vector Machine Detector
PCA	Principal Component Analysis
MCD	Minimum Covariance Determinant
SOD	Subspace Outlier Selection
SOS	Stochastic Outlier Selection
AIVAD	Accurate and Interpretable Video Anomaly Detection
CFA	Coupled-hypersphere-based Feature Adaptation
CFLOW	Conditional Normalizing Flows
CSFLOW	Cross-Scale-Flows
DRÆM	Discriminatively Trained Reconstruction Anomaly Embedding Model
DFKDE	Deep Feature Kernel Density Estimation
DFM	Deep Feature Modelling

DSR	Dual Subspace Re-Projection
FRE	Feature Reconstruction Error
GANomaly	Generative Adversarial Network for Anomaly Detection
PaDiM	Patch Distribution Modeling
RKDE	Region-Based Kernel Density Estimation
STFPM	Student-Teacher Feature Pyramid Matching
UFLOW	U-shaped Normalizing Flow
WIN_CLIP	Windowed-Based Contrastive Language-Image Pre-training
LLaVa	Large Language and Vision Assistant
CLIP	Contrastive Language-Image Pre-training

Chapter 1
Introduction

In this chapter, we give an introduction to the ginger plant plantation and discuss the issues that are associated with the current systems used for the health monitoring of plants. Manual inspection for identifying plant health is generally slow, labour-intensive and prone to human errors making the response to a disease outbreak slow. We are motivated by the necessity to optimize these processes by creating a better system with the aid of machine learning. Specifically, we would like to introduce an unsupervised AD models that might always observe the ginger plants for health issues in detail and at an early stage. It also seeks to improve agricultural practices to provide farmers with better ways and means of detecting health compared to traditional reliance on human labour and knowledge. It also advances the domain of machine learning by showcasing how it can solve real-life problems in the real world of agriculture.

1.1 Ginger Plantation

The ginger plant is a widely cultivated spice all over the world, ginger plant is normally planted in southeast Asia. It is famous for its strong spicy odour and has been credited with several health benefits [1]. The Ginger plant belongs to the Zingiberaceae family. Ginger is a plant, and the preparation involves using the rootstalk to cut the growth of the plant. The method of harvesting ginger is by pulling up the rhizome from the ground, washing it then exposing it to the sun for drying. [2] Ginger plant requires a lot of effort in farming and the introduction of various technology has helped in the plantation. Water and nutrients are pumped to the plant with the help of a pump which is located at a central point and the farmer knows the condition of the plant by looking at it. [3][4] For that reason, it is difficult to transfer this experience to new workers. Based on this, our project is to solve the problem by developing a ginger plant health monitoring system through training in unsupervised AD. It is advisable to have a library that can identify anomalies in plants in a relatively short time using images from the drone as a dataset, using realistic data.

Figure 1.1.1 Ginger Plant [5]

Figure 1.1.2 Ginger Plantation [6]

1.2 Problem Statement

Engaging in the construction of a completely automatized system for the monitoring of plant health is not particularly without difficulties, especially if one needs to apply unsupervised AD towards the task. The challenges highlighted above are what our project aims at trying to address in plant health monitoring. Different from supervised approaches, where the data set contains labelled data, unsupervised AD does not require labelled data and works well in situations where anomalous cases are rare. Additional challenges are observed when working with actual plants in natural conditions including plant size, plant development, surrounding environment, and noise. These factors add more challenges such as achieving detection accuracy regardless of the plant growth health, reducing the false positive rate due to the fluctuations in plant growth, and finally achieving real-time performance despite random recording environments. Our work positions itself to address these challenges and to offer an efficient and sound solution for automated plant health monitoring using unsupervised machine learning.

1. **Human monitoring**

   These days, the process of planting ginger plants calls for the higher experiences eye of the farmer to determine whether or not the plant is receiving the right amount of nutrients. This is why the eye of a trained farmer is vital so that the yield of ginger plants can be nurtured effectively to produce a higher yield of ginger rootstalk. The defect or unhealthy plant is usually discarded thereby increasing losses to the farmer. Therefore, giving the required quantity of nutrients will reduce the loss incurred by the

farmer. The farmer is also required to check which plant is defective and removed so that the farmer does not spend resources on a plant that will not produce a sellable product. Consequently, this entailed a high entry barrier for the green farmer to venture into ginger production through the establishment of their new plantation. Our project wants to solve this problem by aiding the farmer's vision of detecting defective plants from all the healthy plants in the plantation using an unsupervised AD machine learning model.

Figure 1.2.1 [7]
Yellow Spot on Ginger Plant Leaves

Figure 1.2.2
Caterpillar Found in Ginger Plant

2. Real-world monitoring system

Most studies on health monitoring systems concern themselves with hypothetical models of examining real-world settings, for instance, highly close-up images of plant leaves. However, this approach is not feasible in most real-world plantations because it time time-consuming for close-up shots of various plants. Even the environment such as background and noise might contribute to the algorithm results. Most of the current research in this area applies well-defined backgrounds to reduce background noise in the collected data set; however, this is not feasible in a real-world scenario as the nature of soil for a certain point in the field varies from farm to farm, we should take in the background as part of our dataset and find ways to eliminate the vision from taking it into account. In ginger plantations, plants are generally grouped, therefore an effective solution must be one that can identify an anomaly in various plants and isolate the plant

region that has the anomaly from the others. The existing research is mostly concerned with the detection of individual plants, while in practice, the plants may overlap each other, and it distorts the detection algorithms. Following these limitations, our project will train an unsupervised AD algorithm using real-world datasets in hopes of developing a model that can detect anomalies in a real-world plantation environment.

3. **Unsupervised AD**

   The rate of disease incidence in ginger plants is not very frequent; therefore, it is difficult to find a large number of datasets on ginger plant images that are infected, which are essential for the training of the health detection model. Hence our approach entails training an unsupervised AD model using images of only healthy ginger plants. In this way, we will feed the model with samples of what a healthy plant looks like and then be able to train the model on what it means for it to be unhealthy or to point to features that may suggest illness or another defect. This forms a basis for the evolution of a good health -monitoring system that monitors health. The anomalies identified by this model will help for the subsequent models, which one can train for the specific objective of finding health or particular types of diseases in plants. Our project is to devise a dependable approach to unsupervised AD that doesn't only identify the abnormalities in ginger plants, but also gathers and store data on the anomalous plants. It will help extend the current knowledge of the plant anomalies and assist in the future endeavors of developing models specific to the disease.

1.3 Motivation

This project was established because of the rising need to modernize the plantation industry, especially through the application of the latest technology. With the help of the unsupervised AD approach used, the management of the ginger plantations becomes more effective, as the constant monitoring of the plantation's health no longer requires human intervention and the farmer can do other more essential task. The project helps in acquiring valuable information on the health and state of the ginger plants, which is useful not only for health identification but also for the ensuing expansion of the plantation. To the new farmers, the system proves useful because through using it, they are in a position to be provided with well-organized information on how they can start and manage their ginger plantations. In the case of the well-experienced farmers, it helps in the early identification of diseases thus enabling them to

intervene before the situation gets out of hand hence increasing both the quantity and quality of yields. The real-time monitoring makes sure that the farmers get quick feedback on the health of the plants to protect them. Due to the early warning and assessment of the state of plants, the proposed system offers the farmer an opportunity to make the right decisions hence enhancing the yield of the farm and making the farming process less hazardous. This project not only brings the agriculture sector into the modern world but also enhances the abilities of the farmers introducing a new set of tools to regulate productivity and protect the crops from diseases and unfavorable conditions.

Figure 1.3.1 Smart Agriculture System [8]

Apart from the positive impact that this project will have towards the agricultural sector, there is a strong desire to improve the uptake of unsupervised learning, especially AD within the machine learning community. It is for this reason that the true farm data has a lot of variability mainly from the growth stage of plants, environmental effects, and backgrounds that provides a perfect testing ground for machine learning researchers to test how models can be trained and fine-tuned in a real-life environment that is unstructured. As this project is devoted to AD without using any labelled data it is also an attempt to demonstrate what is possible with the help of unsupervised learning methodologies for solving real-world problems. In addition, the efficiency of the system in processing real-time traffic and its expansibility offer important lessons on how to deploy the learning models in applications requiring timeliness and at a large scale. This project be a good reference to the machine learning society, to advance more research on the resilient, flexible models which will be implemented across the other sectors apart from farming.

1.4 Project Scope

The scope of the project is to create a application on the health se monitoring system using a unsupervised AD model training on the detection of ginger plant using the dataset of real life video from a ginger plantation.

1. **Ginger Plant**

   This project goal is to detect AD on the ginger plant using dataset capturing from a drone for ginger plant applications. The project scope is limited only to the detection of anomaly for ginger plant and unsupervised AD model should be produced for this purpose.

2. **Unsupervised AD**

   This project goal is to create a model for the purpose of detecting anomaly in the ginger plant using a unsupervised AD model. The project scope is limited only to the creation of one model.

1.5 Project Objectives

Our project objectives is to create a complete application on the health monitoring system using unsupervised AD. Our project should be able to be used on able to tackle on the real life applications on detection of plant health.

1. **Detection of Anomalous Ginger Plant**

   Our project should be able to detect ginger plant and produce and output on whether a plant is anomalous or normal with very high accuracy. Our detection model should be able to pinpoint the anomalous part in the image and provide the farmer on which plant is problematic.

2. **Practical Health Monitoring System**

   Our project solutions should be able to use real-world situation as a data and achieve good accuracy on detection on ginger plant on actual plantation. Our projects aims to be a application for the use case of detecting health plant in real farm.

3. **Unsupervised AD**

Our project should be detection using a unsupervised AD model using a large sample of healthy plant from the ginger plantation and produce one model of unsupervised AD model

1.6 Contributions

This project aims to contribute significantly to both the agricultural and machine learning communities by providing a robust unsupervised AD model for plant health monitoring. Firstly, it offers a practical, real-world solution for disease detection through the use of real-time camera feeds or video recordings of ginger plants. This approach allows for immediate identification of potential health issues, thereby assisting farmers in taking timely action to manage plant healths and improve overall crop health. By integrating real-time data processing, the project addresses the need for efficient, scalable monitoring systems in agriculture, ultimately aiding in better disease management and increased crop yields.

Secondly, the project seeks to expand the dataset of anomalous plants, which will be invaluable for the future development of computer vision models targeted at health detection. This enhanced dataset will support ongoing research and refinement of AD algorithms, enabling the creation of more accurate and effective models. Additionally, the project includes a user-friendly training module that facilitates the integration of new datasets from different plants, allowing for easy adaptation and training of models on new crops. This feature will assist in the development of plant-specific detection systems and contribute to the broader machine-learning field by demonstrating practical applications of unsupervised learning techniques. Overall, the project not only enhances agricultural practices but also advances machine learning research by providing new data and methodologies for AD in dynamic environments

1.7 Report Organization

This report is structured into seven chapters. Chapter 1 outlines the project background, aim/objectives, problem, and scope. Chapter 2 outlines the review of existing anomaly detection systems in plant disease monitoring in terms of their limitations and the rationale for the approach adopted in this study. Chapter 3 outlines the system methodology in terms of the general approach, data capture, and the model selection process. Chapter 4 outlines the system design, which includes the system architecture, the systems and integration techniques used in developing the system. Chapter 5 outlines the system implementation, including the systems

used and the technologies adopted, data preparation for the data set used for the system, and the preliminary tests performed. Chapter 6 outlines the system evaluation in terms of analysis of experimental results, evaluation of the performance of the model, and the problem encountered. Chapter 7 concludes the report by outlining the conclusion of the study as well as the strengths and weaknesses of the system and areas recommended for improvement in the future.

Chapter 2

Literature Review

2.1 Previous Works on Plant Health Detection

In recent years, the use of computer vision and machine learning models in the detection of health in plants is common, there is various research done on such topic with varying degrees of success on many plants using different models, proving that using such models has the potential of becoming a solution for this issue.

2.1.1 Plant Health Detection using Supervised Learning Model

The most common model used in the detection of health in plants is the CNN model this CNN model used to solve this problem is usually made up of 3 steps, First is the collection of the dataset, training on the model and computer vision techniques to show on the defective area of the plant.

1. Ginger Plant

   Shreelakshmi's paper collected 7014 pictures of label ginger dataset of 150 by 150 pixels images as shown in Figure 2.1.1.1. [9] Shreelakshmi states that CNN is quite effective at classifying the images into healthy and disease categories with high accuracy rates measured from various studies. Table 2.1.1.1 shows the result in the paper where the best method produces a 99.53% accuracy using DCNN, SRGAN, WGAN, VGG16, ResNet50 and DenseNet21 while the other model produces at least 92.68 accuracy.

a) Pest Pattern b) Nutritional deficiency c) Soft rot disease

Figure 2.1.1.1 Label Dataset on Ginger Plant Leaves [9]

Citation	Methods	Techniques	Symptoms	Accuracy
[14]	DCNN, SRGAN, WGAN, VGG16, ResNet50 and DenseNet121	Different stages of GAN	healthy leaves and unhealthy leaves	99.53%
[15]	CNN, ResNet34	F1-Score	definite diagnosis	96.05%
[16]	CINV, RCTs, NVP	Quality Analysis	abdominal pain, bloating, gas, and epigastric distress	94%
[17]	CNN, VGG-16, Inception-Resnet-V2, ResNet-101	Performance Metrics	Brown Spot Disease, Blast Disease, Bacterial Leaf Blight	92.68%
[18]	CNN and ANN	-	Color of the Leaf	96%
[19]	PFP, PPUE	ANOVA	Soil Properties	97.12%
[20]	CNN, MobileNetV2, VGG-16	Deep Learning	Soft, brown, semisoft	98.53%

Table 2.1.1.1 Table on the Result for Shreekakshimi [9]

2. Single Plant Organ Training

   Boulent states that a model that specializes in a single organ of the same plant is better compared to training on a specific plant and therefore better than a model that trains on a variety of plants. [10] The dataset they use is from a public dataset from Plantvillage which contribute some unknown variable as to the time and date or the condition of the image it is taken. The paper uses a CNN model from scratch and due to the limited in data for CNN, it uses previous cycle weightage to continue training. The area of the dataset chosen to be used for training as weed out using feature extraction and fine-tuning. Figure 2.1.1.2 shows the result of their custom CNN model against another model.

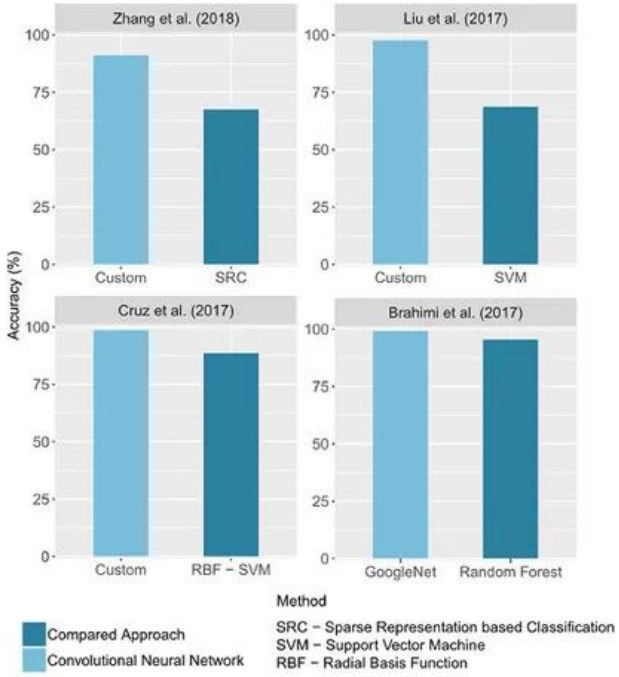


Figure 2.1.1.2 Result of Custom CNN Compare to Other Model [10]

3. Multiple Plant Training

   Geetharamani's paper also uses a dataset provided by PlantVillage accounting for 54305 images of 13 different plant leaves.[11] The datasets are processed and produced in different orientations and provide noise to augment the images. The model they plan to train the Deep CNN model is provided by scikit-learn and other libraries are used to assist in the training such as Keras, pillow and OpenCV libraries. The model design is shown in Figure 2.1.1.3. The images go through convolutional layers that act as a feature extraction to extract defects in plant leaves and perform another layer to leave behind additional discriminative features which are used to build the CNN model on the feature of leaves. The result of this model is used to compare against another model in Figure 2.1.1.4. This proves that the CNN model can be used to classify the health of multiple different plants with high accuracy.

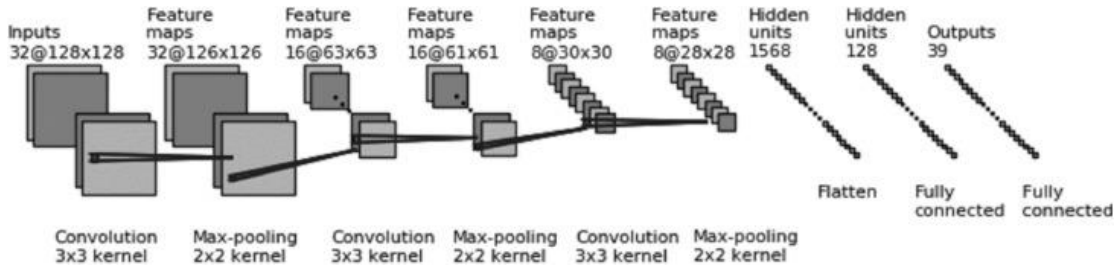


Figure 2.1.1.3 CNN Model Design [11]

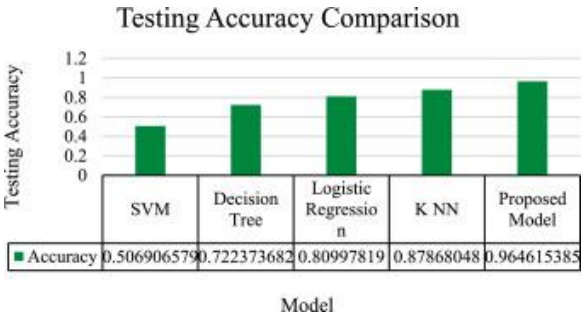


Figure 2.1.1.4 Result Comparing with other Model on Multiple Type of Plant [11]

4. Multiple Model and Fast CNN

   Li's paper works on the need for a workable health detection model on a rice plant.[12] The dataset is collected locally in Anhui, Jiangxi and Hunan Province, China using a handheld phone composed of 1800 images of rice sheath blight, 1760 images of rice stem borer and 1760 images of rice brown spot which is shown in Figure 2.1.1.5. The model they use is a composite of the model. They use Faster-RCNN for object detector in the ROI. A custom DCNN Backbone for the feature detection. Yolo for the detection of the disease. Through a combination of multiple models is able to accurately predict and locate the disease spot from the images.

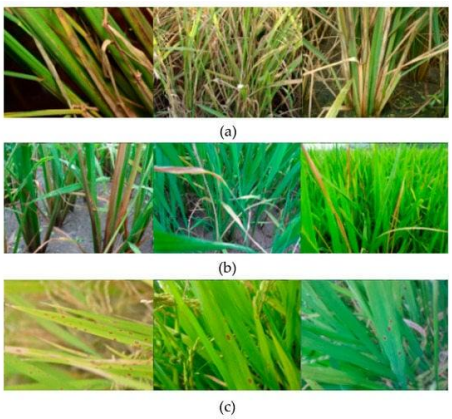


Figure 2.1.1.5 Dataset Images of Rice Plantation [12]

2.1.2 Plant Monitoring System using Unsupervised Learning Model

AD[13] describes a statistical process used in machine learning and data analytics to define atypical patterns, outliers from expected behaviour, or unexpected events within a dataset. This becomes particularly important when applications involving labelled events become too cost-intensive, for instance in fraud detection, cybersecurity, and monitoring within heavy industry. It has attracted significant interest in agriculture because of its ability to detect changes in plant

growth and health, among other features that are hardly visible to the naked eye. Traditional approaches to AD have normally involved some sort of supervised learning model, in which common and not-so-common cases are well-labelled, and thus used for the training of an algorithm. In most practical applications, especially in agriculture, the labelled data is small, and the conditions affecting plant health may differ to such an extent that unsupervised learning models are much more applicable.

In a ginger plantation, the detection of an anomaly will be possibly a very strong tool for real-time monitoring because maintaining yield and quality will depend on the detection of health. Diseases mostly occur in the leaf, stem, and root portions in ginger plants, as in many other crops; therefore, early detection of such diseases is essential to prevent large-scale damage. However, growth characteristics in ginger plants together with environmental factors such as fluctuating weather conditions and uneven soil quality make it difficult to rely on set indications of diseases. In the same vein, unsupervised AD is very helpful, for it does not depend on the extensive labelled dataset either of sick plants but finds deviances from the standard growth habits of healthy ginger plants.

It may further include one field of ginger cultivation, monitoring daily or weekly plant development with the help of real-time video. It could then be analyzed with an unsupervised AD algorithm over time to learn what normal behaviour in plants is under different ambient conditions. This model can identify when a plant or cluster of plants starts to exhibit atypical behaviours like a colour change in leaves, irregular growth, unexpected drooping of leaves, and raise an alert for either disease or other stressor effects. That is, since the model has been trained over general trends within healthy plants, it can handle a wide range of environmental conditions without explicit training in every conceivable disease.

Integration of AD techniques in ginger plant monitoring would provide farmers with early warnings in case there are complications; therefore, they have been able to take their precautionary measures in time before the spreading of the disease all over their crops. This will contribute to more efficiency in the management of the disease and less overreliance on manual surveillance and operations. The models that are not based on prior assumptions flexible for different agricultural environments are scalable and practical for use in identifying crop diseases, for example, ginger.

1. AD for Plant Health

   Adhiwibawa [14] conducted research regarding the use of a citrus plant for AD in leaf health. The dataset that was used in this research consists of 60 images taken by a digital camera. Each image falls into one of two categories: normal leaves and abnormal leaves. Object segmentation is one of the preprocessing steps in which leaves are cropped out from the background to make them the main objects of interest. Preprocessing normalizes the model for AD to only characterize the leaf and not be influenced by any background noise.

   In this regard, AD includes colour feature extraction, which focuses on colour as the main symptom for symptom-based indication of the health status of the leaf. Some diseases or environmental stress often appear as colour changes in leaves, such as yellowing or some darker spots. After that, the colour features are analyzed, and then the system decides if a leaf is normal or abnormal. The AD is done by the T2 Hotelling Multivariate Control Chart, which is a statistical method for monitoring the normal distribution of multivariate data. $T2 = (x - \bar{x})' S\text{-}1(x - \bar{x})$, where x is the sample data, $\bar{x}$ denotes the mean, and S is the covariance matrix. The expression is useful in telling just how far any particular sample has strayed from what had been expected to be the norm.

   For the threshold computation of AD, the F-distribution is computed at a certain significance level (alpha) as Upper Control Limit (UCL) that sets the boundary on what can be an outlier. Samples that fall beyond this threshold are flagged as an anomaly. Figure 2.1.2.1 shows the result of applying the AD model to the citrus plant leaves as a visual representation of the differences between healthy and anomalous samples.

   This technique of AD meets the demand for a strong method that can indicate the possible problem of the citrus plant, whether by infection of diseases or environmental stress, using multivariate data such as colour. The system then gives attention to multiple variables, using statistical control charts that can indeed show the difference between normal variations in the colour of leaves and those which indicate that there is a problem. Techniques such as these become increasingly important for early health detection and agricultural monitoring.

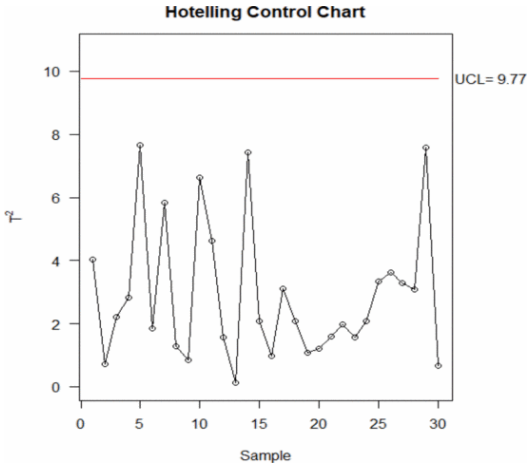


Figure 2.1.2.1 Result of AD on Citrus Leaf [14]

2. AD in Smart Agriculture

Catalano's [15] paper proposes an AD framework for smart agricultural systems using Multivariate Linear Regression and Long Short-Term Memory. He uses Multivariate Linear Regression to analyze the dependent and independent relationship of two variables-like environmental factors and sensor data, respectively a system will use to determine under normal conditions. Meanwhile, LSTM is a powerful model for time series prediction that will enable forecasting future sensor readings based on their historical data. These combinations activate an AD system which will point out any deviation from expected behavior that could be malicious tampering or an environmental issue.

The proposed system provides intrusion detection for intrusion of pests into crops or other humans' attempts to destroy crops. They also detect a cyberattack aimed at tampering with sensor data or causing system failure. This may be facilitated by an AD mechanism analyzing the real-time data from sensors and flagging abnormal patterns indicating harmful events or disruptions. Besides, enhanced security boosts system resilience by promptly identifying system failures or accidents brought about by human error. For example, if a farmer, due to human error, makes a mistake on the plantation, then the AD system would flag this because of inconsistency in the data.

Figure 2.1.2.2: The output of the AD system showing temperature against time. In this context, abnormal fluctuation in temperatures that could indicate tampering with sensors or some other environmental stressor was accurately detected by the system.

Early detection of the anomaly provides valuable insights to farmers about actionable outcomes that help them protect crop yield and system integrity.

It identifies that, in general, AD has become of prime importance in contemporary smart agriculture, since its complexity requires devices that are interconnected and quite impossible to monitor manually for every threat or malfunction that can be imagined. Thus, with advanced LSTM machine learning models automating this process, farmers can manage their operations more efficiently while protecting themselves against both external threats and internal errors.

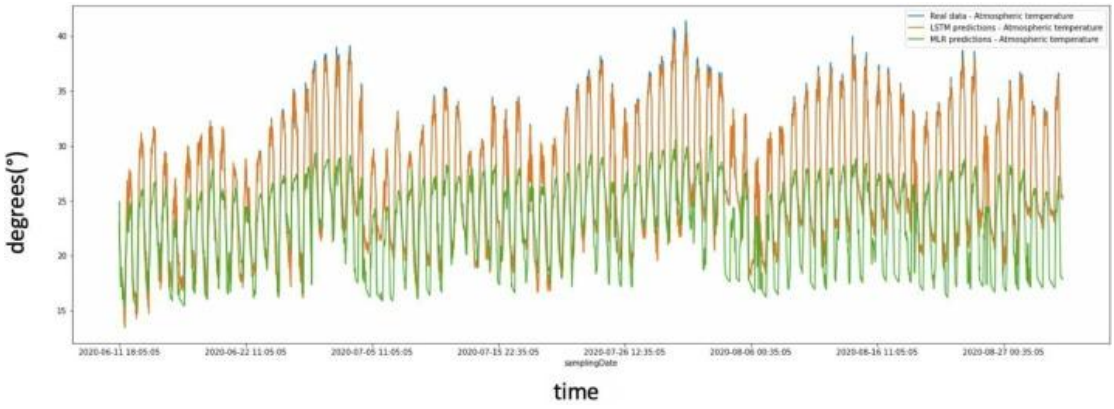


Figure 2.1.2.2 AD Result on Soil Against Time [15]

3. Deep Neural Network AD in Plant

   Deep learning was seen to emerge as a revolutionary technology in agriculture, more in the sphere of plant health detection. This technique had substantial edges over the traditional methods due to the capability of handling large volumes of data, automation of monitoring processes, and enhancement in precision in the identification of early signs of deterioration in plant health. It finds its place in agriculture, where large-scale fields normally cannot be inspected manually. Deep learning has been acting as a game-changing technique in health detection and monitoring crop health based on image analysis.

   Nirmala et al. [16] have discussed the application of deep learning to detect complications in agriculture by using AD. The authors have focused their research on using deep neural networks to classify plants as healthy or diseased by visually identifying features from images taken through sensors placed in the field. In such a

way, this idea integrates deep learning with anomaly-based diagnosis for locating abnormal patterns in growth or health indicative of disease, pest attacks, or other kinds of environmental stressors. Their system applies deep learning to realize a robust real-time, large-scale monitoring of agriculture.

One of the most significant advantages that deep learning models such as DNNs have is analyzing unstructured data-that is, images- and, from them, extracting meaningful features for the identification of health in plants. Unlike the conventional methods that depend on predefined thresholds or manually labelled datasets, deep learning models learn from data input directly. Hence, this can adapt to several types of plants and further environmental conditions. This flexibility would come in handy in agriculture because it takes into consideration different lighting conditions, soil conditions, and weather conditions.

Perhaps scalability is one of the major advantages of using deep learning in plant health detection. Image sensors can monitor large areas constantly, and the deep learning model can work with continuous data to identify deviations from normal plant behavior by real-time processing. Often these minor changes, which a human eye would not notice, might point out very early stages of certain diseases or stress of the plant due to environmental factors and allow farmers to take remedial measures before things take a turn for the worse. With deep learning models, precision agriculture is much needed in areas of accurate and timely data that help maximize crop yields and quality.

Furthermore, deep learning models can be built in such a way that it will automatically recognize various types of diseases for different types of crops, making the application versatile for use across diverse agricultural domains. For instance, based on the colour, texture, or shape of a leaf, the model will be able to classify whether the plant is healthy or diseased and whether it suffers from a nutritional deficiency. It is this capability of handling multivariate data that enables complex patterns to be detected, which otherwise might not be readily apparent with traditional methods of analysis.

Table 2.1.2.1 shows the result of the Deep learning model compare against with other model on the performance evaluation of image processing analysis.

Algorithm	*Accuracy*	*Recall*	*f1-score*	*Precision*
KNN	97	1	1	0.99
Naïve Bayes	98	1	1	0.99
Random Forest	99.1	1	1	1
SVM	96.5	1	1	0.97
Decision Tree	97.5	1	1	1

Table 2.1.2.1 Performance of Model against Other Model [16]

2.2 Limitation of Previous Studies

Most of the earlier works conducted on plant health monitoring are eminently limited and point to the disconnection between theoretically related research and actual application. One of the main problems with most of the studies is the nature of their idealistic approach, which, for the greater part of it, fails to capture or even come close to the complexity and realities of the actual agricultural settings. For instance, most of the related works make do with a dataset comprising no more than a few labelled diseased plants. As pointed out by J. As Boulent et al.[9] pointed out, since these labelled datasets will not only be very labour-intensive but on one hand also impracticable on a large scale, the labelling process is extremely resource-consuming to label each instance of plant disease manually. This approach is unworkable for farmers who are already stretched thin with daily responsibilities. Similarly, labelling every plant for research purposes is just as unrealistic considering the number of plants and the amount of work that would entail.

Most images in such studies are close-up shots or cropped to show only a few spots of the diseases. While such a system serves to simplify the training of models, it does not accurately

portray conditions that exist in real life. In practice, crops are rarely evaluated under optimal conditions. Instead, farmers seek applications that handle and process entire fields of crops with ease in a single run. Although helpful in the controlled environment of a research setup, close-up images cannot present the full dimensions of an actual farm setting where plants are grown in close groups and the backgrounds can be very different. This creates some issues in that the models while training on such images, have difficulty generalizing when applied to more complex, real-world environments.

Another critical limitation regards many of these works failing to take into consideration the dynamic nature evident in agricultural environments. Real farms go through changes in weather conditions, plant growth stages, and different backgrounds contributing to the look of the plants and possible disease manifestations. Training these models on idealized static images omits variances in these factors, hence leading to poor performance of the models when such varying conditions occur. A system useful for practical application therefore needs to rest on rapid, complete field scans, rather than cropped and isolated images. Models should be robust to detect anomalies based on broader plant features and adapt to the variability characterizing real farming conditions.

In other words, much of the available literature related to the monitoring of health in plants is a bit too idealistic, with little consideration or modelling of what happens in the real world with agricultural concerns. This is further exacerbated by the limited, manually intensive nature of dataset labelling, often focused on close-up and cropped images; both factors together detract from the effectiveness that might be achieved in a real-world setting. This underlines the urgent need for solutions that can fill the gap by processing volumes of diverse data representative of real-world conditions to generate actionable insights for farmers in these complex, variable conditions.

2.3 Possible Solutions

This limitation is addressed in this project by designing a scheme whereby real-world images and scenarios are taken for training and testing purposes of AD. Such traditional methods usually rely on idealized data sets, which might not depict the actual scene for such agricultural environmental conditions. Our work overcomes this limitation by taking in video footage of

real life from ginger plantations at various periods. This will ensure that the developed models are not only efficient in laboratory conditions but also in real-life applications.

Real images were used in our project to make the model more robust against the challenges of relatively diverse and dynamic agricultural setups. The video sequences were preprocessed with sequences of motion blur and background changes. This allows our model to learn from a dataset that exhibits realistic properties. This ensures that the idealistic datasets problem is surmounted, as the model learns from images that have real variation in a normal farm setting—from stages of plant growth to light and environmental noise.

For this, we would like to apply methods of AD, for which the model is to be trained on the concept of what a normal ginger plant looks like. That would turn out to be useful in enabling the model to pick up features that deviate from the learned norm, such as symptoms of disease. We are also introducing video footage from a real farm to take into account the real-world motion blur and other artefacts not often taken into consideration within the research setting.

The drones will also capture images of the ginger plantation in high resolution. Drones can fly quickly over large areas, taking several pictures from several angles. It makes the dataset collected rich and variable for training. In this respect, data gathering speeds up and provides the opportunity to consider real plantation variability size in one dataset. It is within this line of thought that our proposal shall, therefore, aim to bring together theoretical research and practical application in suggesting a more reliable and accurate plant health monitoring system. Based on actual data collected in the field, we are evolving a model that should work brilliantly under both artificial and actual-life situations in a step-up mode. This will help in contributing further strategies toward the effective management of diseases at a larger level in agriculture.

2.3.1 Model Options

Anomaly detection models are highly varied in methodology and technological basis and each form is adapted to deal with specific properties of data sets. While there might exist models that perform perfectly well on specific data sets, in general, it's assumed that their performance cannot be generalized to all data spaces. This holds primarily due to data distribution variance, complexity in features, environmental noise, and application-specific needs. There exists no

universally best-performing AD model currently; rather, the performance of each model depends on the specific challenges and conditions of the problem under consideration.

Given these factors, the need for a rigorous and methodical evaluation process in the selection of an AD model for a new application becomes a high priority. When the application involves monitoring plant health based on video images of dynamic and uncontrolled fields, the need for this becomes imperative. A methodical process for selecting AD models includes cross-comparisons of multiple AD models tested on the same data to enable objective evaluation. This enables model-specific merits and pitfalls to be determined concerning their ability to generalise outside the conditions under which it has been trained, robustness to noise and artefacts, and ability to detect nuances in anomalies that constitute early indicators of plant decline in health.

Moreover, benchmarking model performance on diverse datasets is also important to ascertain scalability and real-world viability. A well-performing AD model on benchmark datasets—collected usually in a controlled lab environment—could not perform when implemented in the field environment, as data in the field environment would be more complex, dynamic, and unpredictable. For instance, variation in lighting conditions, clutter in the background, plant placement, and natural irregularity in plant growth could all pose enormous challenges not found in controlled environments. For that reason, intensive benchmarking on diverse datasets must be performed to assist in ensuring that the implemented AD model not only performs correctly but also robustly and reliably for real-world farm monitoring.

In a critical evaluation, the review should include a wide range of AD model types so that the optimal solution can be established. In general, AD models fall into three broad categories: conventional models (e.g., statistical models, traditional machine learning algorithms), neural network-based models (e.g., convolutional autoencoders, deep one-class classifiers and generative models), and the newly developed Vision-Language Models (VLMs), which utilize the strong ability to generalize across multimodality through pretraining. These types all have analogous strengths and weaknesses. The conventional models are usually simple to use, easily interpretable, and efficient but may not have optimal performance on highly complex data. The neural network-based models can learn very complex patterns and non-linear relationships but are highly dependent on hyper-parameter tuning and the need for very large data. VLMs are a

new promising direction that can perform zero-shot and few-shot but their performance in the case of domain-specific fields in plant health monitoring is a subject of ongoing work. Thus to select the optimum AD solution for ginger plant anomaly detection in various applications, representative models of each of the three types need to be tested and benchmarked systematically. These tests need to be performed in conditions approximating the deployment environment in question while being careful to monitor metrics that range from detection accuracy to false positive rates to robustness to variability to computational efficiency. It's only by adopting a scrupulous empirical and iterative approach that we can arrive at an effective high-fidelity AD system to facilitate real-time field-level ginger plant monitoring.

1. Traditional Models

Traditional AD models, including OCSVM, Random Forests, and k-NN, are based on statistical and heuristic methods for anomaly detection. Traditional models perform very well when the feature space is well-defined and low-dimensional, with structured data. The interpretability and simplicity of these models make them very suitable for small or less complex datasets. However, their performance considerably deteriorates in high-dimensional or noisy datasets, since it may fail to capture complex patterns or subtle anomalies often present in plant health monitoring datasets.

I. **ABOD -** This is the method to identify outliers for high-dimensional data. Due to the "curse of dimensionality," distances become less meaningful in high-dimensional spaces. The ABOD method overcomes this problem by focusing on angles between distance vectors of points, which are more informative for AD in such spaces. The key idea is that inliers, or normal data points, cluster around and therefore have higher variance in the angles formed by vectors connecting these points, while outliers typically sit further away from these clusters and tend to produce more uniform angles. ABOD effectively distinguishes between inliers and outliers by comparing the angular variance between data points. This approach capitalizes on the fact that inliers have more variability in their angle distributions, whereas outliers have less variability, which allows their more reliable detection in high-dimensional spaces. [17]

II. **CLUSTER -** Clustering-based Local Outlier Detection transforms multivariate outlier detection into univariate by applying statistical tools designed for one dimension. This

is developed using a new metric, δ is the minimum distance needed by a point to enter an area of a higher-density neighbourhood. The local density is computed for each data point based on its distance from a set of points in some neighbouring region. Once transformed, the dataset is treated as univariate, allowing for the application of statistical methods such as Chebyshev's inequality to estimate the proportion of outliers. The method further refines outlier detection by introducing concepts like ($k\sigma\delta Cm +^{-} \delta Cm$) connected and density peak reachability that allow the identification of outliers about clusters and local density peaks. That significantly reduces the computational complexity required, and it can allow quite accurate outlier detection in a high-dimensional dataset. [18]

III. **COF -** COF algorithm uses a kernel function to transform the input feature space and computes connectivity between data points. This technique first maps the data to a new feature space using a kernel function. Then, it determines the k-NN for each data point, constructing a nearest-trail dataset. The algorithm calculates the average distance of chaining, which can be considered a weighted distance for the nearest-neighbour path. The COF for a data point is computed from the ratio of its average distance to the k-NN and the average distance of its neighbours. A greater COF indicates an increased possibility that the point may be considered an outlier. In COF, several kernel functions are Gaussian and polynomial and Operate on the data to embed them in higher-dimensional space for better outlier detection. In such a way, the performance of anomaly detection will be significantly improved within complex data. [19]

IV. **HISTOGRAM** - Histogram-based Outlier Score algorithm: uses histogram representations of the features' distribution to calculate the likelihood of occurrence of each value by means of the frequency of this very value. First, the histograms are scaled to have a maximum height of 1; the probability of each value is mapped to a scale from 0 to 1. Values closer to 0 are more anomalous. For computing the anomaly score, combine the histogram of all features by taking, for each feature, the logarithm of the inverse probability. A higher anomaly score indicates a higher likelihood of the data point being an outlier. The algorithm also introduces the concept of dynamic histograms, where bins are adjusted in width based on the data distribution, addressing

issues like unused bins and improving the representation of data with varying densities. [20]

V. **IFOREST** - The Isolation Forest algorithm is one of the AD methods that employ the concept of isolation trees to detect anomalies by isolating data points with shorter path lengths. It works by constructing multiple isolation trees, each of which is an expert in detecting different anomalies. One of the strong points of IFOREST is its ability to work with small sample sizes, which enhances its capacity to isolate anomalies effectively. Unlike methods requiring large sample sizes, IFOREST benefits from sub-sampling (that is, selecting instances at random without replacement) because a smaller sample interferes less with normal points, thus giving better anomaly isolation. Common issues in AD, like swamping-in which the algorithm mistakenly identifies normal points as anomalies-and masking, in which anomalies are hidden by dense normal clusters-are handled in IFOREST through partial models. This sub-sampling strategy allows IFOREST to handle these challenges more effectively, resulting in improved detection of anomalies, especially in datasets with large, dense normal point clusters. [21]

VI. **k-NN** - k-NN AD is a distance-based anomaly detection algorithm that identifies anomalous data points based on their distances from other data points in the dataset. Every data point is assigned an anomaly score, which is the sum of distances to its k nearest neighbours. The basic assumption is that normal data points are clustered densely while anomalous points are far away from the majority. The algorithm relies on the choice of k and the distance metric, often Euclidean distance, to determine the nearest neighbours. One challenge of k-NN is its dependency on the selection of k, which can lead to instability, especially in high-dimensional spaces where data sparsity and the curse of dimensionality can degrade performance. To enhance stability and robustness, ensemble methods combine outputs from multiple k-NN models, aggregating their anomaly scores using techniques like averaging or maximization. This helps mitigate issues like parameter sensitivity and variability in unsupervised learning scenarios.[22]

VII. **LOF** - The Local Outlier Factor is a density-based AD that detects outliers by comparing the local density of a data point with the local densities of its neighbours.

Unlike most methods that consider the general distribution of data, LOF concerns the local neighbourhood around a given point. It calculates the Local Reachability Density(LRD), which is the density of a point in its neighbourhood, considering the distances to its nearest neighbours. A point that has a low LRD compared to its neighbours is flagged as an outlier. The LOF score is computed then by comparing the LRD of a point with the average LRD of its neighbours. A LOF score greater than 1 indicates the point is an outlier, with values increasing as the degree of outlierness rises. This method is particularly useful for detecting anomalies in datasets with varying densities and is applied in domains such as medical records, where outliers can represent important abnormalities.[23]

VIII. **MCD** - Minimum Covariance Determinant is a robust statistical technique that finds a subset of data points that has the minimum determinant of their covariance matrix to detect outliers in high-dimensional datasets. MCD focuses on a subset of data of size M, with $\lceil N/2 \rceil \leq M \leq N$, which is least affected by outliers. For any subset, compute the empirical mean and covariance based on the points, and retain the subset with the smallest determinant of the covariance matrix. This ensures that the resulting mean and covariance matrix are resistant to the presence of outliers. MCD differentiates between vertical outliers data points that substantially deviate from the linear relationship but with non-outlying predictors and leverage points, which are points with extreme values for the predictors. While traditional methods for computing location and scatter rely on the least squares residuals and can therefore be insensitive to certain types of outliers, the MCD approach focuses on minimizing the covariance determinant. [24]

IX. **SOD** - Subspace Outlier Detection represents a robust algorithm for finding outliers in meaningful subspaces of high-dimensional data space. It differs from traditional methods evaluating outliers over the full dimensionality of a dataset because SOD performs outlier detection by focusing its attention on subsets of attributes or arbitrarily oriented subspaces. SOD is especially good in high-dimensional spaces wherein the notion of distance or density becomes less reliable by analyzing deviations in localized or lower-dimensional projections. In spatial data contexts, neighbourhood structure based on specific spatial attributes is often utilized, while deviation relative to these neighbours is evaluated using another attribute. This dual focus enables the detection of anomalies that are contextually significant, such as spatial outliers deviating in a

specific feature within their spatial proximity. This approach is widely applicable in areas like geospatial analysis, social networks, and complex multidimensional datasets.[25]

X. **SOS** - Stochastic Outlier Selection is a probabilistic approach toward outlier detection in a dataset. Unlike traditional methods that consider rigid thresholds or distance measures, SOS defines an affinity-based framework to compute the outlier probability of each data point. It computes affinities by comparing pairwise similarities among data points, emphasizing the local neighbourhood structure. It models the likelihood that each point is an outlier through a stochastic process, defining the outlier probability as a result of the relative affinity of a point to its neighbours. Hence, points having low affinity to their neighbourhood will have higher outlier probabilities to set them apart from the rest. SOS is particularly suited for datasets with complex or nonlinear structures, as it does not assume any specific distribution or global data properties. Due to its probabilistic nature, SOS gives a nuanced view of outlierness and hence is of value in applications such as AD, fraud detection, and exploratory data analysis. [26]

2. Neural Network-Based Models

Neural network-based AD models rely on deep learning to represent complex patterns and dependencies inherent in the data. Well-known models in this class are PatchCore, PaDiM, and EfficientAD. These models work extremely well with high-dimensional and unstructured data, such as images or videos, which makes them perfectly suitable for plant health monitoring using real-time video recordings. They use feature extraction from pre-trained networks, localized anomaly scoring, and dimensionality reduction to detect anomalies with high precision. However, most of these models require a lot of computational resources and large amounts of data to train the models, which may not be feasible in resource-constrained environments.

I. **AIVAD -** Accurate and Interpretable Video Anomaly Detection is a robust framework for detecting anomalies in video sequences by combining interpretable and deep features. The approach includes three stages: **pre-processing, feature extraction, and density estimation**. Pre-processing includes optical flow maps that

track the motion of objects across frames, and an object detector localizes and classifies objects within bounding boxes. These feature-extracting objects are represented through a **velocity feature** derived from optical flow to capture motion, through **pose features** based on body landmarks, or other **deep features** extracted from video and images using a pre-trained CLIP model that captures nuances lost in the other representations. For density estimation, anomaly scores are computed with GMMs for low-dimensional velocity features and k-NN models for high-dimensional pose and deep features. These scores are integrated into a unified anomaly measure, thus enabling high detection accuracy along with interpretable outputs suitable for various datasets. [27]

II. **CFA -** The Coupled-hypersphere-based Feature Adaptation method addresses the bias of pre-trained CNNs by refining anomaly localization through feature adaptation and memory bank optimization. CFA learns patch descriptors from normal samples, clustering these features densely around memorized representations stored in a memory bank. Utilizing a hypersphere-based loss function, CFA ensures that normal features are tightly clustered while abnormal features are effectively separated. Hard negative samples in turn provide contrastive supervision to further enhance feature discrimination. The memory bank is then iteratively refined using an EMA for efficient compression of normal feature representations. During testing, CFA compares patch features to their nearest memorized counterparts, generating anomaly heatmaps. A sophisticated scoring function is used that combines distance and certainty metrics to mitigate the underestimation of normal features. This ensures precise anomaly localization, as evidenced by superior AUROC scores on benchmark datasets like MVTec AD, demonstrating its effectiveness in challenging anomaly detection scenarios.[28]

III. **CFLOW** - The method for anomaly detection based on feature extraction and likelihood estimation. Feature vectors containing semantic information are extracted by the CFLOW encoder using a discriminatively-trained CNN with a multi-scale feature pyramid pooling. Since it is pre-trained on large datasets such as ImageNet, both the local and global information is captured to deal with variability in the size and shape of anomalies. The CFLOW decoder estimates the likelihood

of these feature vectors using conditional normalizing flows, integrating spatial priors through positional encoding. Each decoder layer combines feature vectors with spatial conditions in an efficient manner using translation-equivariant architectures. Training entails maximizing the log-likelihood of features, while testing calculates likelihoods and maps anomaly scores via normalized probabilities. CFLOW-AD is computationally efficient compared to methods like Spatial Pyramid Anomaly Detection Embedding (SPADE) and PaDiM, avoiding large memory needs by not relying on train galleries or extensive covariance matrices. It excels in balancing accuracy and resource usage in real-world anomaly detection scenarios.[29]

IV. **CSFLOW** - Cross-Scale Flows: It is one of the advanced techniques that were performed for defect detection on images with a cross-scale flow architecture, allowing the information from feature maps at multiple scales to flow across. By the sharing of information among different scales, this method increases density estimation and likelihood computation. Convolutional layers are applied to the extraction of features, while the features are then furthered into coupling blocks, each performing an affine transformation. These blocks split the feature tensors, apply scale and shift transformations, and recombine, thereby enabling the model to capture complex dependencies across scales. To stabilize training, a soft clamping technique is applied to the scaling components. The aim is to maximize the likelihood of the features by minimizing the negative log-likelihood in a latent space. A key advantage of this method is its ability to localize defects by preserving positional information, allowing defect detection at specific image regions, which is crucial for fine-grained analysis in real-world applications.[30]

V. **DRÆM** - Discriminatively Trained Reconstruction Anomaly Embedding Model combines two key sub-networks: a reconstructive sub-network and a discriminative sub-network. The reconstructive sub-network uses an encoder-decoder architecture to detect and reconstruct anomalies while preserving non-anomalous regions of the image. It is trained to reconstruct the original image from a corrupted version generated by a simulator, with losses combining SSIM and L2 loss. The discriminative sub-network has a U-Net-like architecture, learning a joint

reconstruction-anomaly embedding. It generates anomaly segmentation maps by analyzing the difference between the reconstructed and original images. The approach utilizes a simulated anomaly generation process whereby artificial anomalies are added to the image using random texture augmentations and noise generation. This provides a diverse set of anomalous samples without requiring real anomaly data. This allows the network to learn the representation of anomalies regarding their deviation from normality. The final output is an image-level anomaly score based on the segmentation mask. [31]

VI. **DFKDE** - Deep Feature Kernel Density Estimation: This method incorporates deep learning into kernel density estimation for better performance in complicated scenarios, such as photon mapping. DFKDE leverages a pre-trained neural network backbone to extract the hierarchy of features from the input data that captures the essential patterns and context. Noise is reduced, and computational complexity is lowered for these features by using Principal Component Analysis. The normalization makes the features consistent. This is followed by the modelling of a probability density function using KDE. These learned features are used by the KDE to assign a density score. At training time, KDE fits on normal data to learn the distribution; during inference, regions with low density signify anomalies or deviations. For photon mapping, this approach enables photon density estimation, which is efficient due to encoding of individual photon characteristics and the local context in a learned kernel. It reduces the number of photons needed for achieving accurate results for such effects as caustics and is computationally efficient, which provides fine quality. [32]

VII. **DFM** - This is a deep neural network-based approach that models class-conditional probability distributions in the feature space using the feature representations learned by a DNN. A DNN is trained to classify samples into N classes, and class-conditional distributions are fitted to the deep features at different network layers. It works by computing the log-likelihood of features concerning these distributions at test time for identifying in-distribution samples as high likelihood and, respectively, out-of-distribution or adversarial samples as low likelihood. Unlike previous methods which are based on the assumptions of tied covariance Gaussian

distributions (such as LDA), more flexible models are used by DFM, such as separate multivariate Gaussians or Gaussian Mixture Models, which have a greater ability to capture complex structures of high-dimensional feature space. The parameters of these distributions are estimated by maximum likelihood, while GMM complexity is controlled by criteria including BIC. This generative modelling improves the estimation of uncertainty and robustness in classification. [33]

VIII. **DSR** - Dual Subspace Re-Projection is a novel architecture for surface anomaly detection, taking advantage of quantized latent space representation and dual decoders. DSR processes the input image into two quantized feature maps (high and low resolution) using a ResNet-based encoder and a vector quantization (VQ) mechanism. These two decoders are then specialized into different tasks: the general appearance decoder reconstructs high-fidelity natural images, while the object-specific decoder constrains reconstructions to normal appearances of the trained object. The anomalies are found through the comparison of these decoders' outputs via an anomaly detection module that creates feature resolution segmentation masks, which have been upsampled for pixel-level localization. Apart from that, DSR embodies an anomaly generation method by which its training is performed with the injection of near-in-distribution anomalies in a quantized feature space. It has been trained hierarchically, first in a general image reconstruction scenario, and later on fine-tuning for anomaly detection using a high-resolution upsampling module.[34]

IX. **EfficientAD** - EfficientAD is a lightweight framework for video anomaly detection that aims at reducing the computational cost of the process without compromising detection accuracy. It introduces **Patch Description Network (PDN)**, a shallow convolutional network of only four layers, generating feature vectors for 33×33 pixel patches in one forward pass. To further make PDN efficient and expressive, it is distilled from a deeper pre-trained network such as WideResNet-101. EfficientAD adopts a **lightweight Student-Teacher (S-T) model**, in which the teacher is a PDN and the student imitates the outputs of the teacher, guided by a hard feature loss promoting high-loss regions for training. This enhances anomaly detection by focusing on key features while suppressing false positives. Logical

anomalies are detected using an autoencoder that compares its reconstructions with teacher outputs. Normalized anomaly maps from the S–T model and autoencoder are combined to capture both structural and logical anomalies. EfficientAD can deliver real-time performance with robust localization: less than 1ms on modern GPUs. [35]

X. **FastFlow** - FastFlow is an unsupervised anomaly detection method that seeks to localize and detect anomalies in visual data. The method is based on a representation approach whereby it extracts features of normal images for building a probability distribution and then compares features of test images to this distribution for anomaly detection. The feature extractor has either ResNet or Vision Transformers (ViT). In any case, both ViT are better in capturing the local and global relationships. For ResNet, the features from the last layers of the first three blocks are taken. FastFlow uses a 2D flow model with invertible transformations to map image features into the distribution of normal images. This model measures the likelihood of an image's features by a bijective mapping, where anomalies have low likelihoods. The flow model is constructed using many transformation blocks, each with an affine coupling layer and neural networks. Spatial information is preserved by incorporating 2D convolutions into the flow model, ensuring accurate anomaly localization.[36]

XI. **FRE** -: Feature Reconstruction Error method identifies anomalies by analyzing features extracted from a pre-trained deep neural network (DNN) through a shallow linear autoencoder. It compresses high-dimensional features into a lower-dimensional space and reconstructs them by computing the FRE, which is the difference between the original and reconstructed features. FRE provides an uncertainty score for anomaly detection and localization; larger errors indicate anomalies. Image-level detection relies on the norm of FRE, while pixel-level localization maps FRE errors across spatial dimensions. FRE is flexible; it works with intermediate DNN layers and combines multi-layer FRE maps for enhanced performance. The implementation strategies include PCA-based deterministic methods, which provide computational stability and finite convergence, and iterative autoencoder approaches, more suitable for large datasets due to their

memory efficiency and flexibility in regularization. FRE maps offer precise segmentation by resizing to input resolution, enabling accurate anomaly detection and localization across diverse tasks.[37]

XII. **GANomaly -** GANomaly is an anomaly detection framework that works by training Generative Adversarial Networks on normal data to identify outliers. The model contains three sub-networks: a generator, an encoder, and a discriminator. The generator encodes the input data into the latent space, reconstructs it via a decoder, and produces realistic outputs. A second encoder re-encodes these outputs, and their latent representations are compared to detect anomalies. Training involves three losses: adversarial loss, which makes the generated images indistinguishable from the real data distribution; contextual loss, which minimizes the reconstruction error; and encoder loss, which aligns the latent representations. At inference time, anomalies are scored with the encoder loss since both the generator and encoder fail to represent anomalous features as they were trained only on normal data. GANomaly excels in unsupervised scenarios, offering robust performance for image anomaly detection and localization by exploiting discrepancies in the learned latent space.[38]

XIII. **PaDiM -** PaDiM is a deep anomaly detection model that utilizes the strength of pre-trained CNNs to extract semantic and spatial features for anomaly localization. During training, each image patch is mapped to embedding vectors, which are derived from activation maps of different CNN layers and capture multi-resolution contextual information. These embeddings are modeled as multivariate Gaussian distributions and their mean and covariance matrices are estimated for each patch position across the training data. This approach encodes the distribution of normal data, including inter-layer correlations. At inference, embeddings of test image patches are compared against the learned Gaussian distributions using the Mahalanobis distance that quantifies the deviation of a patch from normalcy. Then it generates an anomaly map where high scores indicate anomalous regions. The method avoids the computational overhead of k-NN-based approaches by using precomputed parametric distributions, making it efficient for both training and

testing. PaDiM excels in anomaly localization and is scalable for high-resolution inputs.[39]

XIV. **PatchCore** - PatchCore is an effective technique for the detection and localization of anomalies by using patch-level features extracted from a pre-trained CNN. The approach was developed by building a memory bank of the localized features aggregated from the intermediate layers of the CNN. Contrary to global feature representations, PatchCore relies on mid-level features to maintain the spatial resolution and contextual relevance required for anomaly detection. In particular, scalability is addressed by PatchCore by employing a greedy subsampling strategy, corset reduction, that retains the diversity and coverage of the memory bank but greatly reduces its size. This reduces computational overhead at inference time with limited degradation in detection accuracy. During inference, every test image is divided into patches whose features are matched to the memory bank through nearest-neighbour matching. Anomalies are identified based on the distance between test patches and their closest nominal patches. A re-weighting mechanism further enhances robustness by adjusting scores based on the rarity of neighbouring patches. PatchCore efficiently generates pixel-level anomaly maps while maintaining state-of-the-art performance across diverse datasets.[40]

XV. **Reverse Distillation** - Reverse Distillation proposes a teacher-student architecture for anomaly detection and localization. The pre-trained **teacher encoder (E)** extracts multi-scale representations, and the **student decoder (D)** reconstructs these features from a compact embedding created by a trainable **one-class bottleneck embedding module (OCBE)**. The OCBE condenses rich high-dimensional teacher features into low-dimensional space with a focus on anomaly-free patterns, mitigating redundancy. It is designed to fail in reconstructing the anomalous features, leading to a **high discrepancy** between teacher and student outputs in case of anomalies. At inference time, **cosine similarity** maps between the teacher and student representations are used to compute both pixel-wise and sample-level anomaly scores, where higher discrepancies would imply abnormalities. The main novelties include **reverse knowledge distillation order**, transferring high-level teacher features to low-level student layers, multiscale features for better anomaly

localization, and compact OCBEs, which amplify the feature differences on anomalies. The experiments conducted on the MVTec dataset show state-of-the-art performance, with particular attention to anomaly localization and detection.[41]

XVI. **RKDE** - The Region-Based Kernel Density Estimation method presents a three-stage anomaly detection pipeline inspired by Hinami et al. These are **region extraction**, **feature extraction**, and **density estimation**. **Region Extraction**: While Hinami et al. used GOP and MOP methods, the proposed method here, RKDE, relies on Faster-RCNN with a ResNet50 backbone. RKDE produces scattered regions which are target-focused using non-maximum suppression. This reduces the number of regions to tens in one frame and hence removes the need to use many normality models. **Feature extraction**: The features are drawn from AlexNet's fully connected (7th layer) or convolutional layers (3rd layer). The fully connected variant accordingly tunes AlexNet for multi-task classification, while the convolution variant maps the features directly to spatial positions, thereby allowing higher processing rates. **Density estimation**: Gaussian KDE is applied on reduced feature dimensions compressed by PCA. Unlike Hinami, RKDE scales feature vectors instead of normalizing them, preserving their length information for enhanced anomaly detection. To address performance, RKDE introduces feature rejection during exposure for the convolutional variant, limiting accepted features to that novel relative to prior data, ensuring efficiency in both analysis and model creation.[42]

XVII. **STFPM** - The Student-Teacher Feature Pyramid Matching framework identifies anomalies using a student-teacher learning approach combined with a feature pyramid structure. The teacher network, pre-trained on a classification task, such as ResNet-18, generates multi-scale feature maps that capture low-level (textures, edges) and high-level (contextual) information. The identically architected student network is trained to emulate the teacher's feature outputs for normal images. The training minimizes differences, using 2-normalized vector distances between corresponding feature maps at each pyramid scale, focusing on hierarchical learning for better anomaly detection across various object sizes. During testing, the teacher and student extract features from an input image, and deviations from their feature

maps indicate the position of anomalies. These deviations get aggregated into an anomaly map that highlights the anomalous pixels. By combining feature pyramids with pixel-level analysis, STFPM provides precise, multi-scale anomaly localization, making it suitable for detecting diverse and subtle anomalies efficiently.[43]

XVIII. **UFlow** - UFlow is a method for unsupervised anomaly detection and segmentation composed of four phases: **Feature Extraction**, **U-shaped Normalizing Flow**, **Anomaly Score Map Generation**, and **A Contrario Anomaly Segmentation**. During the first phase, multi-scale features are extracted using a novel MS-CaIT architecture that combines Transformer models with U-Net-inspired designs for rich, scale-specific feature representations. It will introduce a U-shaped Normalizing Flow architecture for anomaly detection, using invertible transformations that ensure statistical independence of multi-scale embeddings for precise likelihood estimation of anomalies. Then, it calculates anomaly scores, associates a likelihood with each pixel by using these embeddings, and efficiently generates anomaly maps. Finally, the fourth phase applies a contrario framework to segment anomalies, using a hierarchical tree structure of connected components based on level sets of the anomaly map. This ensures accurate anomaly segmentation with unsupervised statistical thresholds while maintaining flexibility and robustness across datasets.[44]

3. VLMs

The recent breakthroughs in AD are based on VLMs that combine vision and language through large-scale pretraining. These models, such as VLLM, leverage both visual and contextual information to improve the performance of AD. Using multimodal embeddings, VLMs can contextualize visual data with semantic information, making them quite powerful for complex scenarios. For example, a VLM can detect anomalies in plant health by analyzing visual cues together with textual descriptions of how plants should look. While these models have immense potential, their actual deployment requires sophisticated infrastructure and careful fine-tuning to effectively address domain-specific challenges.

I. **WinCLIP** – WinCLIP and its extension, WinCLIP+, are state-of-the-art frameworks of zero-shot anomaly classification and segmentation based on powerful language-image embeddings developed for CLIP. WinCLIP improves the detection of anomalies by incorporating **Compositional Prompt Ensembles** (CPE), which describe "normal" and "anomalous" states of objects through structured language prompts in a task-specific way and align text-image representations to increase accuracy in the classification process. Segmentation in WinCLIP follows a **window-based approach** for dense, multiscale visual feature extraction, which allows pixel-level anomaly detection with computational efficiency. It harmonically aggregates predictions across overlapping windows and scales to balance local details and global context. WinCLIP+ extends this by incorporating a few-shot reference association module, using a small set of normal images to create reference memories at multiple scales. This integration improves the detection of anomalies that are context-dependent or hard to define through language alone. By combining predictions from language-guided and visual-based approaches, WinCLIP+ achieves superior performance in both classification and segmentation tasks.[45]

II. **LLaVa** - The proposed LLaVA-o1 framework extends the reasoning capability of VLMs through a structured, step-by-step reasoning process. It involves four stages of reasoning: **Summary**, **Caption**, **Reasoning**, and **Conclusion**, which allow progressive development from problem interpretation to answer generation. Each stage has an embedded tag that will be self-activated by the model without prompt engineering. A key innovation is the **stage-level beam search** during inference, which maximizes reasoning accuracy by selecting the best response at each stage. Trained on the LLaVA-o1-100k dataset, a curated collection integrating general-purpose and science-focused VQA datasets, the model is fine-tuned to improve reasoning and scalability. Results indicate that LLaVA-o1 outperforms baseline models in logical reasoning, instance reasoning, and domain-specific tasks, particularly in math and science. Its structured output design ensures robust, accurate answers and better

performance across complex reasoning benchmarks, demonstrating its efficiency and adaptability.[46]

III. **Phi3** - The Phi-3 model family is a suite of transformer-based decoder architectures optimized for efficiency and performance across different sizes and applications. The **Phi-3-mini** model has a hidden dimension of 3072, 32 attention heads, and 32 layers, with a 4K default context length extendable to 128K using LongRope. It is pre-trained on 3.3T tokens, supports 4-bit quantization for mobile deployment, and leverages the Llama-2 tokenizer for compatibility. The **Phi-3-small**, with 7B parameters, improves multilingual tokenization via tiktoken, allows for an 8192 context length, and features several innovations, including blocksparse attention, which helps to improve training and inference speeds while reducing KV cache requirements. The **Phi-3.5-MoE** uses a **Mixture-of-Experts** architecture where, out of 16 expert networks, two are activated at any one time to achieve efficiency. Training is focused on high-value, filtered data that emphasize reasoning and logical competencies rather than low-value information. Post-training involves **supervised fine-tuning** (SFT) and **direct preference optimization** (DPO), enhancing safety, reasoning, and user interaction capabilities. These models showcase adaptability, efficiency, and on-device performance, suitable for both research and practical deployment.[47]

IV. **Llama3** - The Llama 3 model family builds on the dense Transformer architecture of its predecessors, Llama and Llama 2, focusing on improved data quality, diversity, and training scale rather than architectural overhauls. Key enhancements include **Grouped Query Attention** (GQA) with 8 key-value heads for faster inference and reduced key-value cache sizes, and an attention mask that isolates self-attention within documents, particularly effective in long-sequence pre-training. Llama 3 has a 128K tokenizer (28K more than last time, and entirely covers non-English languages), with better compression rates and multilingualism without hurting English performance. The model uses **Rotary Positional Embeddings** with a higher base frequency of 500,000 which allows for context lengths of up to 32,768. Architectural details differ by size:

the largest, 405B model, has 126 layers, a hidden dimension of 16,384, and 128 attention heads. Training achieves near-compute-optimal scaling, utilizing 3.8 × 10^{25} FLOPs, ensuring robust performance across diverse tasks.[48]

V. **Moondream** - Moondream is a 1.6-billion-parameter language model developed by Vikhyatk, combining the works of **SigLIP** Phi-1.5 with the vast training dataset of LLaVa. This model has been developed with research applications in mind, strictly for non-commercial use, hence being an important asset in the academic world. The architecture represents the blend of state-of-the-art techniques combined with quality datasets, thus showcasing a commitment to advancing the frontier of AI. Moondream's objective is to establish a touchstone for efficient computation, innovation, and artificial intelligence development, reinforcing its role as a significant tool for advancing the field. [49]

VI. **BakLLaVa** - BakLLaVA is a VLM developed through a collaboration between LAION, Ontocord, and Skunkworks AI. Based on the Mistral 7B base model, this VLM has been expanded by the LLaVA 1.5 architecture for further improvement. Designed for efficiency, BakLLaVA integrates smoothly with the llama.cpp framework, offering developers a faster and more resource-friendly alternative to models like GPT-4 with Vision capabilities. This innovation underlines its great potential for real-time applications and scenarios where computational resources are limited, hence making it very powerful for visual and language understanding. [49]

VII. **MiniCPM** - MiniCPM focuses on the most efficient training of small language models to be deployed rapidly on end devices using Model Wind Tunnel Experiments (MWTE). Three key aspects that MWTE probes are hyper-parameter scaling, optimal batch size, and learning rate stability. The model leverages Tensor Program techniques for stabilizing scaling across different model sizes and thus improves loss prediction accuracy without extensive tuning for each scale. Experiments on batch size show that optimal batch sizes minimize loss while balancing resource utilization, using a modified approach

from Kaplan et al. MiniCPM's experiments on learning rate show that a consistent learning rate of 0.01, despite model scaling, yields the lowest loss across different model sizes. These insights into hyper-parameters, batch size, and learning rates enable efficient training and scaling of SLMs, providing a foundation for larger models, improving both computational efficiency and model performance during training, and ensuring that optimal parameters can be applied to LLMs as well.[50]

The category choice for the model depends on several aspects, such as the complexity of the dataset, available resources, and desired accuracy. Classic models are simple and interpretable; neural network-based models bring state-of-the-art performance to complex visual data, while VLMs represent the frontier of AD by combining vision and language to gain a nuanced understanding of anomalies. This will surely call for a wide-based review of models across these categories to ensure the most feasible solution for plant health monitoring systems.

2.4 Summary

Literature reviews emphasize that previous works on plant health monitoring were suffering from significant limitations, mainly idealized approaches that do not match real-world agricultural environments. Such challenges include reliance on small manually labeled datasets and the use of close-up or cropped images that limit the applicability of these methods in dynamic farm settings. Additionally, existing models often overlook environmental variations like weather, plant growth stages, and diverse backgrounds, which are critical for real-world robustness.

In this respect, the proposed project adopts an AD approach using real-world video footage from ginger plantations. Training models with images of healthy plants would allow them to detect deviations from learned patterns, thus finding anomalies that would enable early disease detection without requiring a large amount of labelled data. Video-based data captures realistic conditions, including motion blur and environmental variability, enhancing the model's robustness and applicability in practical farming scenarios.

The review also points out that suitable AD detection models should be selected by comprehensive evaluation. AD models can be divided into traditional methods, neural network-based approaches, and advanced VLM. Traditional models are simple and interpretable but have difficulties in high-dimensional data. Neural network-based models do well in processing

complex visual data but require a lot of computation resources and big training data. VLMs are the latest development in the field, integrating vision and language to allow for contextual AD, but require highly sophisticated infrastructure.

This project combines some of these approaches to lay emphasis on practical applicability and robust model evaluation in an attempt to bridge the gap between theoretical research and practical implementation of plant health monitoring.

Chapter 3

System Methodology/Approach

This chapter gives an overview of the overall methodology adopted to design and implement the plant health anomaly detection system. It summarizes the significant steps adopted for developing the system, which vary from data acquisition and processing to model creation, system integration, and output generation. Each module of the system—such as the anomaly detection model, database, API, and user interface—is explained in terms of its aim, functionality, and communication with other modules. The methodology further explains tools, frameworks, and methods used to guarantee the system is scalable, correct, and fit for agricultural applications in reality. The scientific methodology spells out how the system was produced step by step.

3.1 System Overview

The proposed plant health anomaly monitoring system here is deployed in a modular manner, where one module can run independently but part of a whole pipeline. Four primary modules form the system: Anomaly Detection Model, Database Module, API Module, and User Interface (UI). A block-by-block design facilitates flexibility, maintainability, and scalability.

1. **Anomaly Detection Model**

   The Anomaly Detection Model is the core analytical engine of the system. It accepts input plant images or video frames, extracts features, and applies an unsupervised learning algorithm to detect anomalies that may indicate plant health or disease. The model can detect subtle, out-of-distribution variations without labelled data.

2. **Database Module**

   The Database Module is responsible for storing all useful information, raw input videos or images, and processed results (e.g., heatmaps, anomaly scores) as well as user information, and system logs. It maintains structured and manageable information to be used for analysis, audit, and potential future retraining.

3. **API Module**

The API Module is the interface layer with the backend system and external components, including the UI and third-party services. The API Module provides RESTful endpoints for operations such as submission of data, fetching the detection results, and system status inquiry. Interoperability and integration into other agri-monitoring platforms are made possible by the modular API design.

4. **User Interface Module**

   The User Interface (UI) Module is a user-friendly platform on which users can engage with the system. Upload plant data, run anomaly detection processes, and view output results in the form of heatmaps and anomaly scores. The interface is designed for simplicity and accessibility to address the needs of technical and non-technical users.

These modules combined provide a robust platform that can monitor plant health in real-time or batch-wise, enabling early plant anomaly detection and proactive crop management.

3.2 Data Collection and Preparation

The data set used in this study was built from real-world observations of ginger plants, amassed over an extended growing period and prepared for the purpose of unsupervised anomaly detection. The preparation is explained in a series of key steps, as follows.

1. **Data Acquisition**

   Data was accumulated in collaboration with a local ginger farmer over the period from week 3 to week 20 of the plant growth cycle. Video images were taken from three angles: side, 60-degree angle, and top-down. The different angles were intended to capture the visual structure of the plants in a more integrated way, so that subtle changes in health, not easily noticeable from a single angle, can be determined. To maintain temporal consistency, video frames were sampled at a fixed rate of one frame every three seconds, which yielded images in JPG or PNG format.

2. **Image Labeling and Annotation**

   Pilot studies suggested the need for a small labelled dataset to enable evaluation and guided preprocessing. Four of the key growth stages—weeks 3, 8, 12, and 18—were selected for annotation. Two types of labelling were used:

- Rectangular bounding boxes for plant pot localization.
- Segmentation masks to isolate individual plants, with the first row of plants targeted consistently for consistency.

Annotations were saved in YOLO format before being reorganized through renaming and restructuring to suit the system's expected input pipeline.

3. **Dataset Variants and Resolution Handling**

   Two dataset variants were developed during preprocessing:

   - Variant A: Entire images were resized to square dimensions. This approach, though, resulted in too much pixel loss, particularly rendering fine features of plants less discernible.
   - Variant B: Original resolution was kept. Segmentation mask-based cropping was performed to remove background objects and enforce focus on the plant object. Week-by-week cropping was also performed to normalize plant size and position within the dataset. This variant was selected for further processing and model training.

4. **Masking and Background Removal**

   Segmentation masks were used to generate plant-only images by removing the background. This helped in training the anomaly detection model on the plant region alone, with less noise from soil, pots, or surroundings. These masked images formed the core dataset that was utilized for training and testing.

5. **Data Augmentation and Anomaly Simulation**

   To generate a more varied dataset and simulate potential plant anomalies, the following augmentation techniques were employed:

   - Contrast variations: Reducing and enhancing contrast levels to mimic lighting anomalies.
   - Color channel adjustments: Three sets of random colour variations to mimic discolouration or pigment loss.
   - Grayscale conversion: Used to mimic sensor degradation or environmental stress.

- Hue shifting: Small upward and downward hue shifts to mimic chlorosis or abnormal leaf colouration.

These augmentations expanded the dataset and mimicked a range of real-world conditions that might not be present in the original data.

This structured and multi-stage data collection and preparation process yielded a high-quality, diverse dataset for the development and evaluation of unsupervised anomaly detection models in realistic agricultural settings.

3.3 Anomaly Detection Model

Three prominent approaches were explored for evaluating the performance of different anomaly detection methods in ginger plant health monitoring: traditional machine learning, deep learning-based models, and vision-language models. Each approach was developed and tested based on performance, hardware feasibility, and appropriateness for the nature of the dataset collected. Below is a summary of the methodology, results, and ultimate model selection.

1. **Traditional Machine Learning with PyCaret**

   The first approach utilized PyCaret, an open-source low-code machine learning library that provides functionality to explore a broad set of classical models with ease and speed. Input features were colour- and statistical-based descriptors that were being extracted at the image level. While PyCaret permitted easy testing and iteration, the results were different, and performance metrics would alternate between precise and imprecise classification. The lack of contextual and spatial information in traditional models limited their ability to detect complex plant anomalies, leading to approximately 50/50 accuracy on test sets. Due to this variability, PyCaret-based models were not included in the final deployment.

2. **Deep Learning-Based Models with Anomalib**

   The second approach utilized Anomalib, an open-source deep-learning library developed by Intel for industrial-strength anomaly detection. Anomalib offers a collection of pre-trained models tailored to unsupervised anomaly detection with normal-only training data. Training is done by providing only healthy (normal) plant

images as input to the model, while testing includes both normal and anomalous samples. 50% of good and bad samples were randomly selected for testing to verify each model's performance in distinguishing between in-distribution (normal) and out-of-distribution (anomalous) inputs.

Among the models listed in Anomalib, the following five were shortlisted based on promising initial performance:

- CFLOW
- FASTFLOW
- PatchCore
- Reverse Distillation
- STFPM

Each model has specific strengths and weaknesses with varied capabilities of anomaly detection and localization. Their disparity in performance is governed by low-level architectural design choices and feature extraction processes. To further boost general detection accuracy and robustness, the input dataset was again optimized through selective pruning as well as carefully engineered augmentation procedures. These changes aimed at aligning the data with each working mode of each model more appropriately. A detailed explanation of the working mechanism of each model is given in the Literature Review.

3. **Vision-Language Models with Ollama**

   The third approach tested the use of vision-language models (VLMs) with the Ollama system, which facilitates the use of light, pre-trained AI models on local devices. The aim was to employ semantic comprehension to detect anomalies that are not necessarily visually evident but can be inferred contextually. The output was inconsistent and usually unpredictable, however, owing to hardware limitations and the generic nature of pre-trained models at hand. Similar to the traditional approach, model output showed approximately 50% reliability in detecting real plant anomalies. VLMs were thus not selected for deployment.

To recapitulate, deep models were chosen based on Anomalib to be resistant, scalable, and yield confident outputs in a manner of being unsupervised. Utilizing the models, trained on

data samples of the healthy, only allows the system to mark structurally or visually defective plants as diseased confidently, without rigorous labelling of disease status. An evaluation and final selection process with an end were established to ensure the final choice of model structure abides by the practical demands of real life along with agricultural surveillance goals.

3.4 Evaluation Metrics and Testing Conditions

This outlines the testing and evaluation scenarios that have been used to assess the performance of the anomaly detection models. It includes the training configuration, such as the loss functions, stopping criteria, and output metrics like AUROC and AUPR. In addition to typical quantitative metrics, there was also manual inspection for proper anomaly interpretation. The testing set was defined to reflect the visual variation that happens in the real world through specific augmentations and the removal of uninformative samples. Readers can anticipate a complete description of how model output was quantified, interpreted, and authenticated within the carefully constructed setting.

1. **Model Training Configuration**

   The same setup was employed to train all models to foster conformity across evaluation. Every model had a maximum of 300 training epochs. Early stopping, nonetheless, was enabled via a patience of 10 epochs and a minimum loss delta of 0.01, allowing the training process to terminate as soon as model performance plateaued. The primary objective function utilized in training was the model-specific loss function from every architecture within Anomalib, tailored to unsupervised anomaly detection tasks.

2. **Quantitative Evaluation Metrics**

   Two main performance metrics were generated by every model:

   - Image-level AUROC (Area Under the Receiver Operating Characteristic Curve): It is utilized to evaluate how well the model can distinguish between normal and abnormal images.
   - Image-level AUPR (Area Under the Precision-Recall Curve): It is utilized to evaluate the precision-recall trade-off, which is of concern in imbalanced datasets.

   To further measure performance, precision, recall, F1-score, and accuracy were also manually computed. These were calculated by observing model predictions at face

value, verifying whether the detected anomalies represented actual visual defects (e.g., disease, stress) or were misdetections of non-anomalous less common features such as background texture, shadows, or pot rims.

3. **Qualitative Assessment**

With the process of unsupervised learning and very few labelled samples of anomalies, human visual analysis was crucial in deciding model outputs. It involved checking whether the anomalies picked by the model had correct visual signatures or else they were artifacts of such situations as background noises or rare objects not seen during training. Images, where the anomaly was falsely detected owing to such situations, were marked as false positives and rectified accordingly. This process enabled better comprehension of the behaviour of each model beyond numeric measures and also needed to compute evaluation metrics such as precision and recall correctly.

4. **Testing Conditions and Dataset Pruning**

Initial attempts at extensive testing across many image variations and metrics were found to generate no helpful results or cause a negative impact on model training and assessment. As a result, the dataset was reduced to remove low-impact or misleading samples that misled the model or added noise to evaluation. Strategic pruning in this manner helped limit the test scope and improve interpretability of findings.

The final test dataset was classified in the following controlled and real-life conditions:

- Normal Samples: Unchanged images, baseline inputs.
- Contrast Adjustments: Contrast adjusted 0.7, 0.9, 1.1, and 1.3 to cover fluctuations in environmental illumination.
- Colour Changes: Artificial distortions representing plant stress at varying rates, grouped under Dying 1, Dying 2, and Dying 3..
- Hue Shifts: Lower hue values of 15 and 30 degrees, creating dry or reddish visual appearances that are generally indicative of disease.

By limiting the test set to such precise conditions, testing became more focused and informative. The refined dataset allowed higher correspondence to application goals under real-world conditions and allowed detection capability assessment to be better and more fairly measured for every model.

In total, model comparison of anomaly detection employed both quantitative metrics and qualitative checking to create a well-rounded model performance. Baseline comparability was established through standard metrics like AUROC and AUPR, but human verification was necessary to contextualize results within agriculture imaging. Redundant or unnecessary data pruning helped improve the reliability of testing, and controlled augmentations helped enable consistency in benchmarking among models. This assessment phase ensured that only models which could detect interpretable and meaningful anomalies were shortlisted to be tuned and further developed.

3.5 Activity Diagram

To provide a clear depiction of the overall workflow in the anomaly detection system, an activity diagram was prepared to illustrate the sequential steps from data collection to anomaly prediction. The activity diagram illustrates the elementary steps undertaken during system development and model deployment.

The process begins with video footage donated by farmers, capturing ginger plants from the side, 60-degree, and top angles for several weeks. The videos are then processed to obtain the image frames every three seconds. The frames so obtained are annotated manually using pot-bounding boxes and segmentation masks for the plants. The annotated images are then stored in YOLO format and named consistently to maintain the dataset intact.

Once annotated, the images are subjected to a series of preprocessing operations. These involve resizing, cropping, background removal through segmentation masks, and image augmentations such as contrast adjustment, colour randomization, hue transformation, and grayscale conversion. These are added to simulate real-world variability and anomalies in plant health.

Once preprocessed, the images are split into training and test sets. The training set contains only normal (healthy) plant images, while the test set contains both normal and anomalous images. The dataset is then fed to selected anomaly detection models where training is performed using unsupervised learning techniques. Training is to be performed up to 300 epochs with early stopping enabled to prevent overfitting.

After model training, the system proceeds to evaluation, where the trained models are evaluated on augmented test images. Evaluation involves computing standard metrics such as the Area Under the Receiver Operating Characteristic (AUROC) and the Area Under the Precision-Recall Curve (AUPR). There are also manual inspections to understand the model's ability to localize or reason correctly anomalies, especially in cases where infrequent visual features (e.g., unusual backgrounds or lighting) may be induced.

The final step is prediction, where the trained model is used to assess new images. The system gives anomaly heatmaps or binary flags that indicate the presence of potential health issues in the plants.

The activity diagram in Figure 3.5.1 nicely summarizes this end-to-end process, to convey the structured methodology utilized in developing and testing the anomaly detection system.

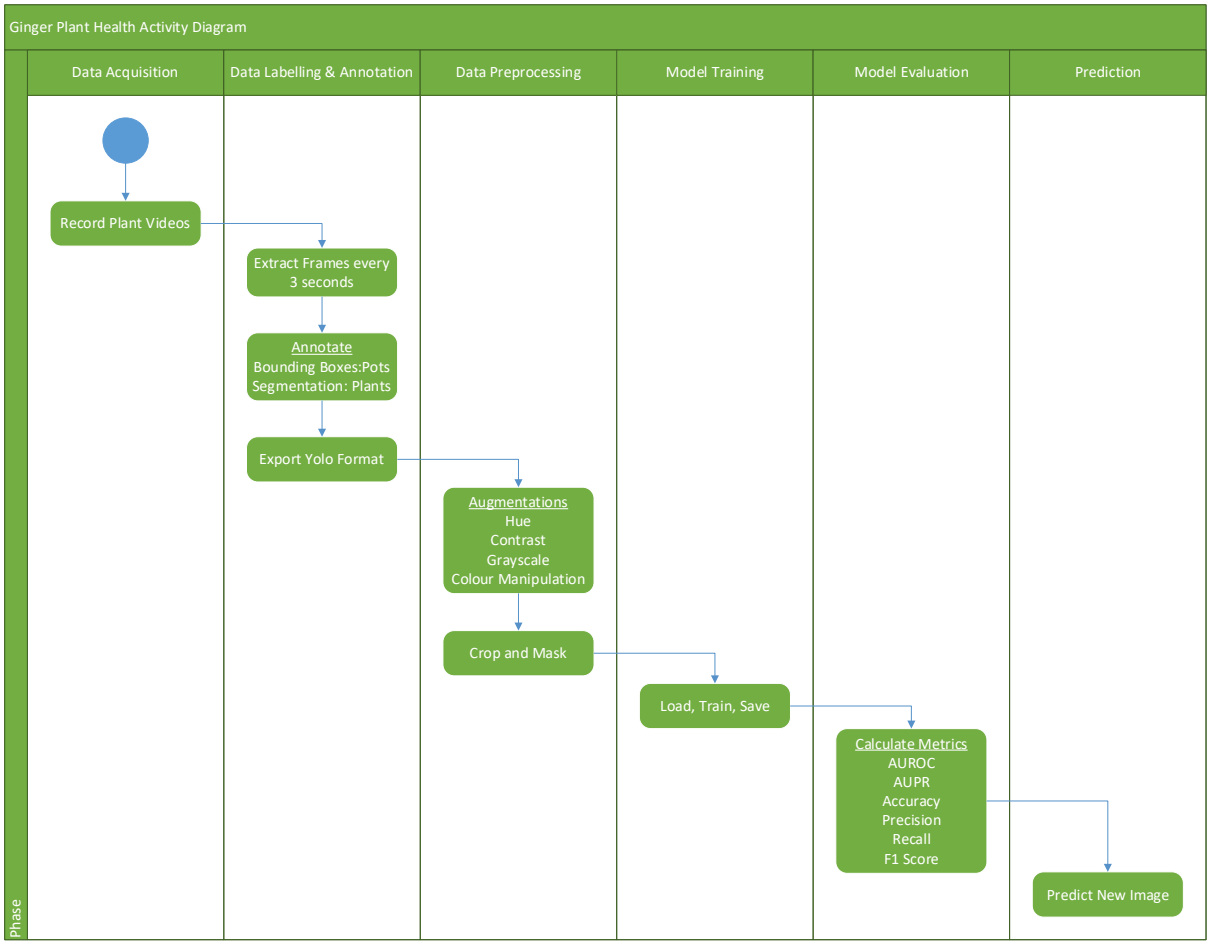


Figure 3.5.1 Activity Diagram for Ginger Plant Health Detection System

Figure X: Activity Diagram showing the process flow of the plant health anomaly detection system, including processes such as video capture, preprocessing, model training, evaluation, and prediction.

3.6 Summary

Briefly, the system methodology outlines an iterative and procedure-based approach towards developing an optimal plant health anomaly detection system. From data gathering and preparation during the initial steps to choosing and evaluating models of anomaly detection, each step was designed to emulate actual farming circumstances while upholding technical detail orientation. Through the utilization of both automated methods and human interpretation, the method ensures not only quantitative performance but also contextual relevance. This foundation provides the foundation for integrating the chosen model into a broader system architecture, which is more fully described in the System Design section.

Chapter 4

System Design

The specification on the project to achieve the problem statement. The project is to build a library on the health monitoring system for the detection of gjnger plant health and disease.

4.1 System Model

This plan of how the software should be operate and how each component works together.

4.1.1 Project Workflow Overview

Figure 4.1.1.1 illustrates the context diagram of the proposed plant health monitoring system. The system consists of two external components and one system component: Ginger Plant, User and the Plant Health Monitoring System. The plant health monitoring system is designed to capture real-time visual data, primarily using image-based inputs to assess and predict the health status of plants. The system leverages unsupervised learning techniques to detect anomalies in plant health by learning visual cues such as changes in leaf color, texture, shape, or other features that may indicate early signs of disease or stress. Once the visual data is processed, the results are transmitted to the user.

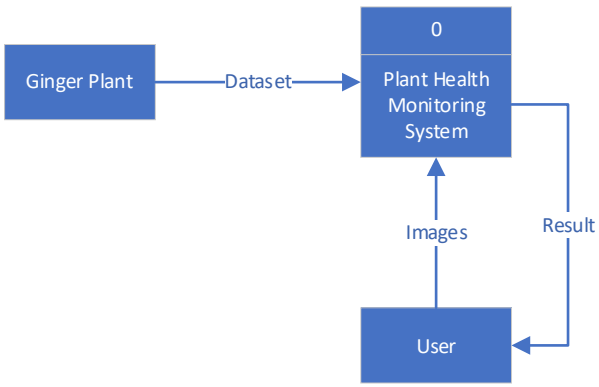


Figure 4.1.1.1 Context Diagram for Plant Monitoring System

There will be a UI to allows users to view real-time results through an application. Users can obtain the health status of their plants and receive alerts if any visual anomalies or potential diseases are detected and the region it occurs.

4.1.2 Full System Component

Figure 4.1.2.1 illustrates the key components of the system that we are required to develop. The system relies on several critical components to function effectively, and these can be divided into three distinct applications, each serving a vital role in the overall workflow.

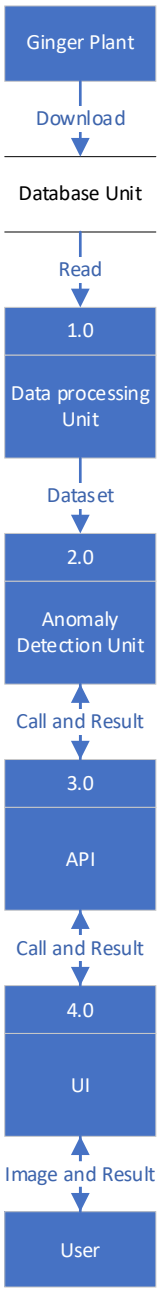


Figure 4.1.2.1 DFD-Level 0 for Plant Monitoring System

The system is integrated with different units that are designed to perform specific tasks and contribute towards the smooth running of the AD pipeline. All these units collectively

contribute to efficient data storage, processing, analysis, and user interaction, hence presenting a complete framework for ginger plant health monitoring.

1. **Database Unit**

   First comes the **Database Unit**, which serves as the project's file storage system. Instead, the unit organizes and stores all datasets, trained models, prediction results, and code in a structured way, not relying on traditional databases. Both historical datasets and real-time data are organized for easy access, and trained models, together with results, are stored for later use and analysis. The efficient organization of resources in this unit supports smooth data flow throughout the system and contributes to its scalability and maintainability.

2. **Data Processing Unit**

   The second unit is the **Data Processing Unit**, which develops and processes raw data coming from the Database Unit into the required format for analysis. It supports various formats depending on different libraries. For example, it can develop the path of images as dictionaries, prepare folder structures suitable to be used in Anomalib or generate Pandas DataFrames for broader compatibility. By handling these preprocessing tasks, the Data Processing Unit ensures that the raw data is prepared for further processing and model training.

3. **Anomaly Detection Unit**

   The core of the system is the **Anomaly Detection Unit**, which hosts the models responsible for detecting anomalies in plant health. This unit manages the entire lifecycle of the AD process: training, predicting, testing, and evaluating the models. It will be integrated with various libraries such as Anomalibto ensure compatibility and flexibility in model selection. This unit also supports hyperparameter tuning and evaluation metrics to optimize the models for accurate and reliable predictions.

4. **API Unit**

   The **API unit** would represent the interface between this external system and the internal AD models. This provides different endpoints related to various functionalities like training models, real-time prediction capabilities, or returning results. The purpose

of the API Unit would thus ensure ease of use as it abstracts the inner process flow of the system when such facilities or functionalities must be shared by other applications or users directly.

5. **User Interface Unit**

   The **UI unit** represents the very last component of the system, which will deliver the processed results to the end user. This unit takes the form of a messaging application, implemented through Discord. Users can utilize the system in real-time: they get the results of predictions and visualizations of detected anomalies right in a Discord channel. The UI will outline the ROI where anomalies are detected, hence making it easy for the user to interpret the output from the system and take appropriate action. This messaging-based approach is accessible and easy to use, considering users who are used to Discord as a platform for communication.

The system will contain six interlinked units:

1. **Database Unit**: used for the storage of structured files.
2. **Data Processing Unit**: loading and formatting of data.
3. **Anomaly Detection Unit**: managing and using AD models.
4. **API Unit**: interaction with the system from outside.
5. **UI Unit**: real-time user interaction over Discord.

All these units together form a cohesive framework that uses machine learning, real-time data processing, and user-friendly interfaces for the effective and efficient monitoring of the health status of ginger plants.

4.2 Requirements Specification

For the system to achieve its objectives of monitoring the health status of ginger plants accurately and efficiently, the following are some of the requirements for each of the six core units:

1. **Database Unit**
   - The system should have a structured file storage system to store datasets, models, results, and code.

- The stored datasets should contain historical and real-time visual data for training and prediction purposes.
- Trained models and results should be saved for reusability and further analysis.
- The system should be able to provide for the effective retrieval and management of data stored therein for smooth integration with other units.

2. Data Processing Unit

- The system should read raw data from the Database Unit and convert it into the required formats: dictionaries of image paths, Anomalib-compatible folder structures, or Pandas DataFrames.
- It should support all typical preprocessing-augmenting, normalization, and resizing to prepare the data for analysis.
- The unit needs to process data of various formats and structures for different machine learning libraries.
- Processed data must be accessible to any subsequent components for training, prediction, and evaluation.

3. Anomaly Detection Unit

- The system shall contain various machine-learning models for detecting anomalies in the health of ginger plants.
- It should support model training, prediction, testing, and evaluation workflows.
- Users must be able to fine-tune model hyperparameters to optimize performance.
- The unit should feed back model accuracy and predictions to the user for the assessment of system performance.
- Models should integrate seamlessly with the Data Processing for efficient operation.

4. API Unit

- The system should include an API that will allow external applications or users to interact with its components.
- The API should provide model training, running predictions, and retrieving results.
- It has to be user-friendly and documented well enough to be easily integrated with other systems.

- Real-time interaction should be supported by the API to enable efficient processing of data and AD.

5. **UI Unit**
    - The system should provide a Discord-based messaging application for real-time user interaction.
    - The users should have the capability to see live prediction results, including segmented images with highlighted ROIs indicating the presence of anomalies.
    - Notifications and alerts in cases of anomaly and disease detection in plants, which may be provided through UI.
    - It should ensure an easy and friendly manner so that users can draw inferences from results and monitor performance.
    - All historical data and other visualizations should be viewed through this UI for necessary decisions to be made.

System Requirements

- The system has to integrate the five units, namely, Database, Data Processing, Anomaly Detection, API, and UI into one unit.
- Every unit should be modular, operating independently for easy debugging.
- All units working together should enable correct real-world operation.
- The system shall deliver effective and accurate health evaluations of the ginger plants for early detection of their anomaly conditions.
- Users must have a responsive and intuitive interface to conduct monitoring and decision-making.

These assure that the system will be robust, modular, and user-oriented to meet the demands of real-world plant health monitoring and AD.

4.3 Modular Class Design and Code Reusability

One of the greatest strengths of the design approach in this system revolves around modular architecture, which makes the code more maintainable, reusable, and adaptable for further modification. All main operations within the pipeline of anomaly detection, ranging from the pre-processing of data to evaluation and notification, have been encapsulated into contained

classes exposing easy-to-use interfaces and hiding implementation aspects. What this achieves is to enable third-party package or model parameter changes by modifying a single class, without having to modify the remainder of the system.

To enable a clean development process and mitigate technical debt, the following class design principles were used:

1. **Layered Abstraction and External Wrappers**

   Third-party tools and services, such as pre-trained models, notification bots, or file system utilities, are wrapped in their classes to conceal third-party logic. A good example is the Discord.py class, which does all the logic required for sending evaluation feedback or anomaly notifications to a Discord server. If the Discord API changes, only this class needs to be changed, and the rest of the system remains functional.

   This abstraction reduces coupling and makes external dependencies modular and substitutable. Certain wrappers may also have dataset converters, augmentation helpers, or model exporters.

2. **Core Pipeline Classes**

   The core pipeline of the anomaly detection system is structured as a set of logically separate and purpose-specific classes, each doing a distinct step of the end-to-end pipeline. This class-based, modular architecture ensures the segregation of duty, well-defined responsibilities, and uniform maintenance across the entire pipeline. Rather than combining multiple functionalities into a single component, the system commits each operation—such as data preparation, model execution, and result analysis—to a distinct specialized module. This means higher cohesion in individual components and reduces the likelihood of injecting hidden dependencies or side effects.

   By segmenting the system into independent and distinct units, maintainability is greatly enhanced through the design. Independent writing, testing, and updates can be performed for each pipeline stage, enhancing debugging and extension. For instance, if a new preprocessing algorithm or model architecture needs to be added, developers can integrate these changes directly into the specific module without refactoring or inspecting other parts

of the codebase. This approach also improves regression testing as changes to one part of the system are less likely to affect others' behaviour.

Separation of concerns also allows for reproducibility and consistency in the performance of the system. The preprocessing stages ensure the standardization of input data before it reaches the model-making process, minimizing variability and curbing the possibility of data leakage or format inconsistencies. Likewise, the model execution module encapsulates training, inference, configuration management, and model storage logic—enabling replacement or tuning of various anomaly detection approaches as desired. The evaluation phase is standalone responsible for the analysis of results, generation of metrics, and detection of cases that need human validation. Every component executes its task in a predictable and controlled way, which leads to a stable and transparent pipeline.

This design strategy not only keeps current development straightforward but also lays the groundwork for future scalability. New models, metrics, or diagnostic tools are easily added by extending or replacing one or more modules. If research advances or system requirements shift, the modular design guarantees that the system will be flexible and resilient. Additionally, such an architecture is ideal for collaborative development, where different team members or contributors can focus on specific pieces without repeating efforts or creating integration problems.

Lastly, clear, well-differentiated core classes ensure that the anomaly detection pipeline is technologically sound and resilient but also easily deployable in the long term in research and real-world environments. It allows the system to naturally develop while possessing a clean, testable, and comprehensible codebase.

3. Unified High-Level Interface for Integration

To allow seamless interaction with the anomaly detection system, one unified high-level interface has been used to capture the complexity of internal functionality. The interface serves as a one-point entry to downstream applications to facilitate integration through activities such as integrating the workflow of anomaly detection into a web dashboard, integration with API-based applications, or embedding within automated pipelines for batch testing and processing.

Instead of relying on developers to acquire and manipulate the internal machinery of each processing module—such as data preprocessing, model setup, training, prediction, and model evaluation—the interface provides directly accessible, reasonable methods that abstractly orchestrate these pieces together behind the scenes. Not only does this keep headloads of developers, especially newcomers to the project team, lower but also reduces error in misuse, ensuring reliability in different modes of deployment.

The interface at the top level is designed to prioritize usability and modularity. By encapsulating low-level interactions with the core pipeline classes inside straightforward function calls, the system allows the common workflows (such as model training, inference on new data, or execution of evaluation routines) to be invoked for with minimal code and without explicit control over low-level settings. Furthermore, the abstraction layer supports pre-defined parameter presets, environment tests, and logging facilities, which enable standardized execution across different environments and use cases.

It also improves maintainability and extensibility. With time, for example, new anomaly detection models, metrics, or notification channels, the joined interface can be added without modifying how the outside systems use it. The attached services and legacy scripts keep running as long as the interface contracts are not being violated, which reduces the possibility of system-wide refactoring or compatibility issues.

Secondarily, the unified interface matters to enable testing and continuous integration. Its modularity makes it easier to be internally mocked for unit or integration testing, and its uniformity allows repeatable automated experiments or benchmarking. This kind of dependability is particularly important in research applications where reproducibility takes centre stage or in production systems where reliability and accuracy come first.

Overall, the integrated high-level interface brings the system from a collection of technical modules and transforms it into a cohesive and usable tool. It ensures that anomaly detection functionality can be easily deployed, scaled, and customized for any number of practical and research-based uses without sacrificing flexibility or control.

4. **Scalability and Future Adaptation**

The system has been thoughtfully designed with scalability and adaptability for the future as its core principles, making it transition easily in light of new demands or technological shifts. From incorporating additional anomaly detection models, employing advanced data augmentation strategies, to using cloud-based storage solutions, the system provides for easy growth with minimal disruption. This flexibility is made possible to a large extent by the modular class design, where new pieces can be added with little or no modification to existing modules.

At the centre of such flexibility is the use of base classes and their adequately documented interfaces. The introduction of new functionality is achieved by inheriting existing base classes, or new logic can be introduced by following the documented patterns within the system architecture. This makes future updates significantly less complicated as the underlying architecture is not meddled with and new components are just added or replaced as and when the need arises. If, for instance, a new, higher-performance anomaly detection model comes out, developers can integrate it smoothly by declaring the correct methods and pointing to it in the existing pipeline without needing to modify other parts of the system.

Also, the system has been designed to support rigorous testing features. Regression tests and unit tests are supported natively, allowing a single module to be tested independently using mock inputs. With this, any modifications done to the system—whether they involve bug fixes, performance improvements, or adding new features—do not inadvertently affect the overall performance. Such isolation support for testing is extremely critical in the role of maintaining reliability as the system becomes larger to handle more data, complex models, or multiple use cases.

This object-oriented modular paradigm not only ensures the integrity of the system increases but also ensures long-term sustainability. The architecture enables seamless incorporation of new research results, tools, or techniques in such a manner that the system remains relevant and adaptable in an evolving field. From a release to a product or the porting of the system to several research environments, the scalable design allows upgrades and additions to be accomplished with as little friction as possible.

Briefly, the system scales and adapts with ease because it is built modular and class-based. It becomes safe for engineers to make improvements, introduce new functionality, or address shifting requirements without fear of destabilizing another part of the system. Future-proofing is achieved by keeping the system stable as it is now while enabling it to be robust enough to handle future challenges or opportunities that may arise.

For detailed code examples, method listings, and structure of each class, refer to the class implementation readme page. [51]

4.4 Software and Packages Used

The below software and packages were selected to implement the anomaly detection system with maximum performance, flexibility, and scalability. Every one of the tools was picked based on its ability to accommodate the specific requirements of the project, from the basic anomaly detection operations to system integration and user notification.

4.4.1 Software

- Python 3.11.9: Python is the primary programming language of the system. Version 3.11.9 was selected due to its performance enhancements, increased stability, and backward compatibility with most machine-learning libraries. The dense library ecosystem combined with the widespread use of Python in data science makes it an ideal platform upon which to develop advanced machine learning and anomaly detection systems. The readability and flexibility of the language also contribute to simplifying development complexity, allowing for easy debugging and tuning through iterative testing cycles.

- Discord: Discord is utilized for notification and communication in the system. It is a notification system in which results of evaluation, anomalies, and status updates are published. Stakeholders are notified in a timely fashion of significant events, for instance, when an evaluation is complete or when there are anomalies. The use of Discord enhances real-time communication and provides an easy-to-use interface for system notifications that improves user experience and system operation efficacy.

4.4.2 Python Packages

- Anomalib: Anomalib is a specialized Python library for anomaly detection that provides a collection of pre-trained models and utilities to facilitate unsupervised anomaly detection on visual data. Anomalib is utilized in this project as the primary tool for model training and inference. It accommodates different state-of-the-art anomaly detection models like STFPM (Spatiotemporal Flow Prediction Model) and PatchCore, which are critical to identifying abnormalities in plant health using video and image data. The adaptability of Anomalib allows the system to easily transition between models, which means one can experiment with different methods of plant health detection and fine-tune configurations depending on performance measures.

    - Model Training and Testing: Modularity in Anomalib allows a straightforward model interchange, so it is possible to experiment with diverse strategies without implementing major modifications into the codebase. As a demonstration, the system can change from the STFPM model to PatchCore, and even between training, testing, and anomaly detection through a simple interface for accommodative adjustment of the system to divergent conditions or specifications.

    - Preprocessing of data: In addition to model-oriented functionality, Anomalib also provides essential preprocessing tools for the data. These include masking, resizing, and augmenting the training image and video data to ensure that the data set is equally prepared and formatted.

- Discord: The Discord library is utilized for providing real-time notifications to a given Discord server, which is a simple and effective way to alert users or stakeholders of critical events, such as model testing completion or detection of anomalies. Discord notifications are especially useful in keeping the team informed of the system's performance without any need for manual intervention.

    - Real-time Alerts: For instance, the system will automatically notify through Discord when it finishes training a model or performing an inference task, alerting users about the completion status and evaluation results such as AUROC scores,

accuracy, or number of anomalies detected. Such integration keeps stakeholders informed at all times regarding the system's performance.

- Integration with Model Evaluation: DiscordNotifier class is closely coupled with the ModelEvaluator component, which handles model output. Once an evaluation has been finished, the system sends a message to a given Discord channel, including a summary of important performance metrics, e.g., accuracy, precision, recall, or AUROC. This is especially useful for monitoring in research or production settings, where timely insight is essential.

- OpenCV: For image and video processing operations in the system, OpenCV was used to handle various preprocessing activities. OpenCV is used for extracting frames from videos, resizing, masking, and segmenting images. OpenCV's flexibility allows the system to efficiently handle image transformations such as applying filters, resizing images, and handling more complex operations like object detection or image enhancement. This is particularly significant in preprocessing steps where raw video recording information is converted to useful input formats for the model. OpenCV's simplicity in integrating with other libraries and complete collection of image manipulation functionality position it as a key component within the pipeline. It also provides the performance that can handle processing big data sets, which is crucial in real-time anomaly detection systems.

4.4.3 Other Considerations

- Scalability and Extensibility: These selected tools are not only useful under the current infrastructure of the system but are very scalable and can be easily extended to support upcoming changes. Any new anomaly detection algorithms or complex methods can be easily incorporated into the existing system by extending the corresponding base classes or adding new logic. For instance, if there is a new anomaly detection algorithm, it can be integrated with minimal disruption to the other components of the system. Modularity afforded by toolkits like Anomalib and Flask means new components can be added without necessarily complicating the system.

- Cross-Compatibility: With the use of Python and the most widely used libraries like Flask and Anomalib, there is an assurance that the system will be compatible with most environments and platforms. The system may be hosted in various servers, deployed in various cloud environments, and augmented with additional tools such as cloud storage for data processing or additional notification services.

These packages and software tools act in concert with each other to form an integrated, agile, and elastic system that can identify anomalies in plant health data. Utilizing established, widely-supported technologies, the system is robust in use and flexible in a way that allows it to extend in the future.

Chapter 5
System Implementation

The system implementation part provides a thorough description of the configuration and deployment process of the anomaly detection system. It outlines the steps needed to set up the project environment, e.g., installing dependencies, installing the necessary software, and combining the various elements used in the pipeline. The implementation process is scheduled to ensure smooth deployment, from preprocessing the data to testing the model, and includes a demonstration of the functioning system with sample input data to display its operational output.

Issues encountered during the implementation phase are also tackled, citing issues such as dependency management, model tuning, and data preprocessing complexities. The issues are solved to ensure the system's robustness and reliability. In addition, the section provides remarks about the overall performance of the system, pointing out locations where future improvements or optimizations would make the system even more scalable and responsive in real use. The objective of this section is to provide an explicit instruction guide for installing, running, and debugging the system, encouraging simplicity and effective deployment to varied environments.

You can refer to the original repository for all the information here.[52]

5.1 Hardware Setup

The development, training, and evaluation of the anomaly detection system were conducted on a local machine with the following hardware specifications:.

- Processor: AMD Ryzen 7 6800H with Radeon Graphics
- Memory: 16 GB DDR5 RAM (4800 MT/s) – 8 GB utilized during model training and inference
- GPU: NVIDIA GeForce RTX 3060 Laptop GPU (6 GB VRAM)
- Storage: 1 TB SSD
- Operating System: Windows 11 (64-bit) x86-64

This hardware setup is a mid-range consumer-grade machine that is capable of handling the computational demands of the anomaly detection system. The large count of cores and multi-threading capability of the AMD Ryzen 7 6800H processor enabled efficient parallel processing of tasks like frame extraction, data augmentation, and preprocessing. As only 8 GB of the 16 GB RAM was used in the model training and inference processes, this setup offered a compromise between computational power and memory, typical for consumer-level hardware.

The inclusion of the NVIDIA GeForce RTX 3060 laptop GPU provided a very welcome performance boost for deep learning tasks. With 6 GB of VRAM, it allowed for faster training and inference of models like PatchCore and STFPM, albeit the GPU capacity remains mid-range relative to higher-end, dedicated equipment. That aside, it performed well in the project's context, handling real-time test scenarios with barely any lag.

The 1 TB SSD provided more than ample storage for the datasets, model weights, and result logs, accelerating overall data retrieval and storage speeds, especially when compared to conventional HDDs. The SSD also enabled managing the project's large video files and numerous model configurations easily.

While the system was adequate for the scope of the project, it remains limited by consumer-grade hardware. The processor and GPU are sufficient for research tasks, but for handling larger datasets, more complex models, or more computationally heavy loads, the higher-spec components (such as additional RAM, a higher-end GPU, or a dedicated server configuration) would perform better. The configuration is ideal for academic and small-scale projects but may lag when extended to more advanced work.

Despite these limitations, the hardware setup at the consumer level was still capable of delivering results within an acceptable timeframe for the sake of this study. Additionally, the system is easily scalable or portable to more powerful cloud-based or high-performance configurations if it needs to be so for future applications, allowing for possibilities of future developments or commercial applications.

5.2 Software Setup

To set up the project environment for development and execution, the following software and packages are required. This includes Python, Git, GitHub Desktop, virtual environment tools, and all necessary dependencies specified in the requirements.txt file. The following steps outline the process for setting up the software environment. ***For more comprehensive information refer to the ReadMe[53].***

1. **Python Installation**

   Ensure that Python 3.11.9 or later is installed on your machine. Python is the core language used for the development of the anomaly detection system, and the system relies on Python's robust ecosystem of libraries to handle data processing, model training, and evaluation.

2. **Git or GitHub Desktop**

   You can clone the repository using Git or GitHub Desktop. Both methods provide an easy way to obtain the latest project version and ensure that you can pull updates as needed.

3. **Creating a Virtual Environment**

   It is recommended to create a virtual environment to keep the dependencies isolated and ensure compatibility with different systems. This is especially useful when working on multiple projects that may require different versions of libraries.

4. **Install Dependencies**

   Once the virtual environment is activated, install all the necessary dependencies using the ***requirements.txt*** file provided in the repository. This ensures that you have the right versions of the required packages.

   This will install all the required Python packages, including essential ones like Anomalib, OpenCV, Flask, and Discord, among others. For a full list of dependencies, refer to Appendix A: Software and Package Requirements.

5. **Verification**

After installation, you can verify that all dependencies are properly set up by running a sample script or checking the installed libraries using:

pip freeze

This will list all the installed packages along with their versions, allowing you to confirm that the setup was successful.

By following these steps, you will have a fully configured environment that is ready for running the anomaly detection system, ensuring that all dependencies are in place for the system's proper operation.

5.3 Setting and Configuration

This section presents all the settings necessary for dataset preparation, system parameter tuning, environment variable management, and integration with the Discord bot. Each part of the system must be well configured to provide real-time interaction with the anomaly detection system during training, testing, and seamless execution. The project is developed with maintainability, reproducibility, and real-world deployment in consideration. This part also talks about compatibility problems and requirements for the system to perform well, especially regarding Anomalib and platform-dependent permissions. ***For more comprehensive information refer to the ReadMe[53].***

5.3.1 Dataset Configuration

To load and execute the anomaly detection model properly, the dataset needs to be organized into a well-formed directory tree. Training and test datasets need to be split into the following folders under the root directory, i.e., datasets and ensure there is train, good and bad folder.

The names good and bad for the directories and train are called out within the ***anomalib_train.py*** file within the TrainObject initialization block.[53]

The dataset is processed by the function DatasetUnit.AnomalibLoadFolder in ***dataset_lib.py***, which is in charge of folder parsing, preprocessing, and loader instantiation. Customizations such as input normalization, resizing strategy, or grayscale support can also be done in that function. Make sure all images are of compatible formats and have the same resolutions to prevent preprocessing errors.

5.3.2 System Configuration

This configuration does not use a "patient epoch" or early-stopping. Instead, it relies on good dataset organization and static settings defined in the training pipeline. These are set primarily through two files: ***anomalib_train.py*** and ***dataset_lib.py***.

- The ***num_workers*** parameter controls concurrency of the DataLoader. A high value can be used on multiprocessor machines to improve data loading performance.

- Training and test batch sizes are also defined in the Folder constructor inside the same method. They can be tuned by GPU memory or RAM constraints, especially when dealing with high image dimensions or deep networks.

- Resizing of images is defined in ***ImageInfoObject*** in anomalib_train.py. Currently, it is 256x256 pixels by default. Ensure that the images in your dataset can be resolved to your preferred size to avoid distortions and memory overflows.

- Due to Windows' multithreading and I/O limitations, training scripts should all run using privileged mode. Failing that, the app might not create worker threads or use system resources that come with reading a file concurrently.

5.3.3 Environment Variables

All bot tokens and webhook URLs used by the Discord bot are stored securely in an ***.env*** file in the root directory of the project. This approach keeps secrets out of source code and prevents accidental exposure.

Create a ***.env*** file and include the following entries:

- TOKEN_BOT_GITHUB=<your-bot-token>
- CHANNEL_WEBHOOK_LOG=<webhook-url-for-log-channel>
- CHANNEL_WEBHOOK_PREDICT=<webhook-url-for-predict-channel>
- CHANNEL_WEBHOOK_DEBUG=<webhook-url-for-debug-channel>
- CHANNEL_WEBHOOK_CLONE=<webhook-url-for-clone-channel>
- TOKEN_BOT_GITHUB is used to authenticate the Discord bot.

Webhook URLs are distributed into particular Discord channels for prediction output, debugging, logging, and cloning commands. ⚠️ Be warned: Never commit or send out your ***.env***. Such tokens and URLs need to be treated as secret.

5.3.4 Discord Bot Setup

1. **Creating a Discord Application**

   To create a Discord application, visit the Discord Developer Portal and log in to your account through Discord. Click the "New Application" button to begin, then enter a name for your application and click "Create." This will begin your project as a Discord application. Once created, you can view all configuration options such as adding a bot, setting OAuth2 permissions, and generating credentials. This application forms the basis for your Discord bot and allows it to join servers, listen to commands, and respond to channels via the API.

2. **Creating Webhooks**

   Create a webhook for a Discord channel by first going into your Discord server and looking at the specified text channel whereby you wish to receive messages coming from the bot. Click on the gear icon next to the channel name to access the Edit Channel settings. Then, go to the Integrations tab and click on Create Webhook. Name your webhook and optionally upload an avatar to make messages sent by the webhook visually recognizable. Copy the Webhook URL created—this URL is important, as it allows your application or bot to send messages directly to the selected channel. Copy this URL into your project file under the corresponding key (e.g., CHANNEL_WEBHOOK_LOG, CHANNEL_WEBHOOK_PREDICT, etc.). Repeat this for each channel where you want to have separate webhooks.

5.4 System Operation

The system operation describes how the anomaly detection system operates interactively through the Discord bot interface. It highlights the initialization process, channel-based command access control, and the overall workflow from model training to anomaly class prediction with visual and colour-coded output. Each stage is accompanied by Discord channels for user communication, logging, and threaded updates in a streamlined manner.

Below is an outline of the major operational aspects, along with representative figures for each process.

1. **System Initialization Configuration**

   The first task in the system operation is to set up the configuration of the Discord webhooks and the respective bot-accessible channels. Since the system employs four distinct webhook channels—log, prediction, debug, and clone—it is essential to restrict particular bot commands to specific channels. This usability and security measure ensures that training commands are not executed in prediction channels, and vice versa. The bot reads the ***.env*** file and associates each webhook with its corresponding Discord channel ID, determining the command set for that channel. This setup also enables modular usage across various servers or projects.

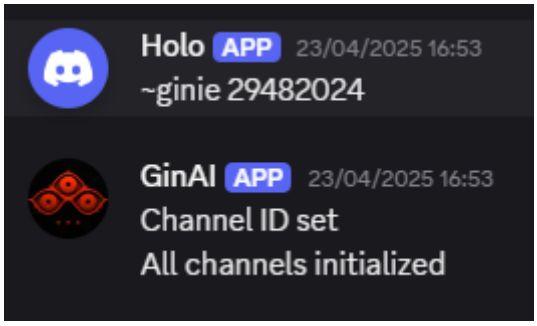


Figure 5.4.1: System Initialization Configuration for the channel-webhook mapping visualization

2. **Training**

   Users initiate model training by sending a training command along with a model name. This command can be executed in the appropriate channel as determined by the bot's channel validation system. In response to this command, the bot acknowledges the request by sending a confirmation message and then logs all relevant training actions (including loading of datasets, epoch progress, and validation performance) to the log channel. Simultaneously, a new Discord thread for that training session is created, where real-time discussion and updates are wrapped and structured. Threaded logging here provides a step-by-step, readable history of training events, model checkpoints, and status messages.

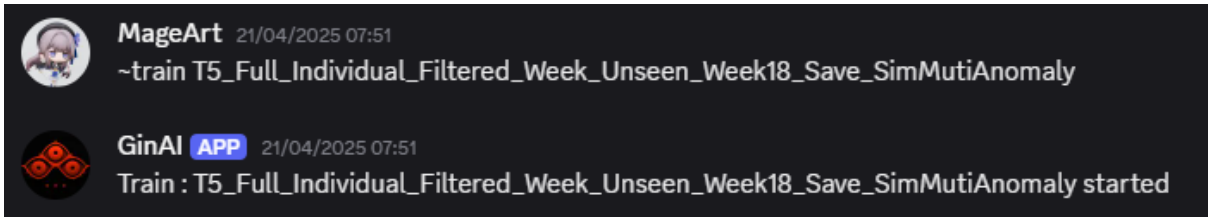


Figure 5.4.2: Train Command Invocation

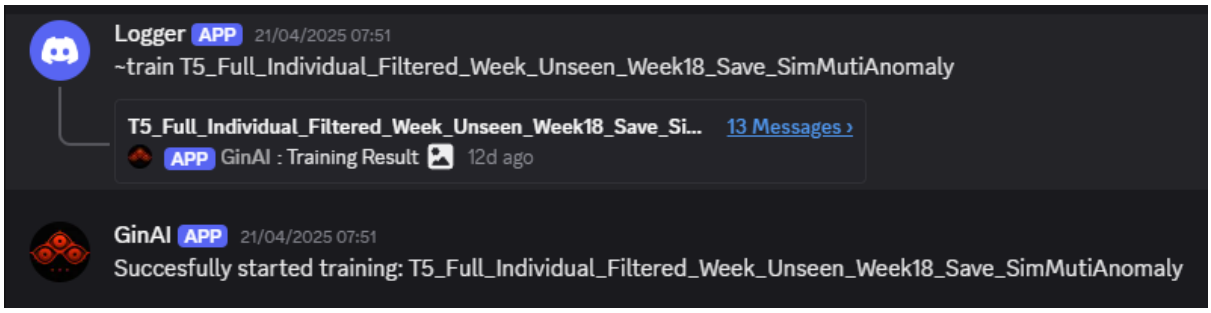


Figure 5.4.3: Log Channel Confirmation

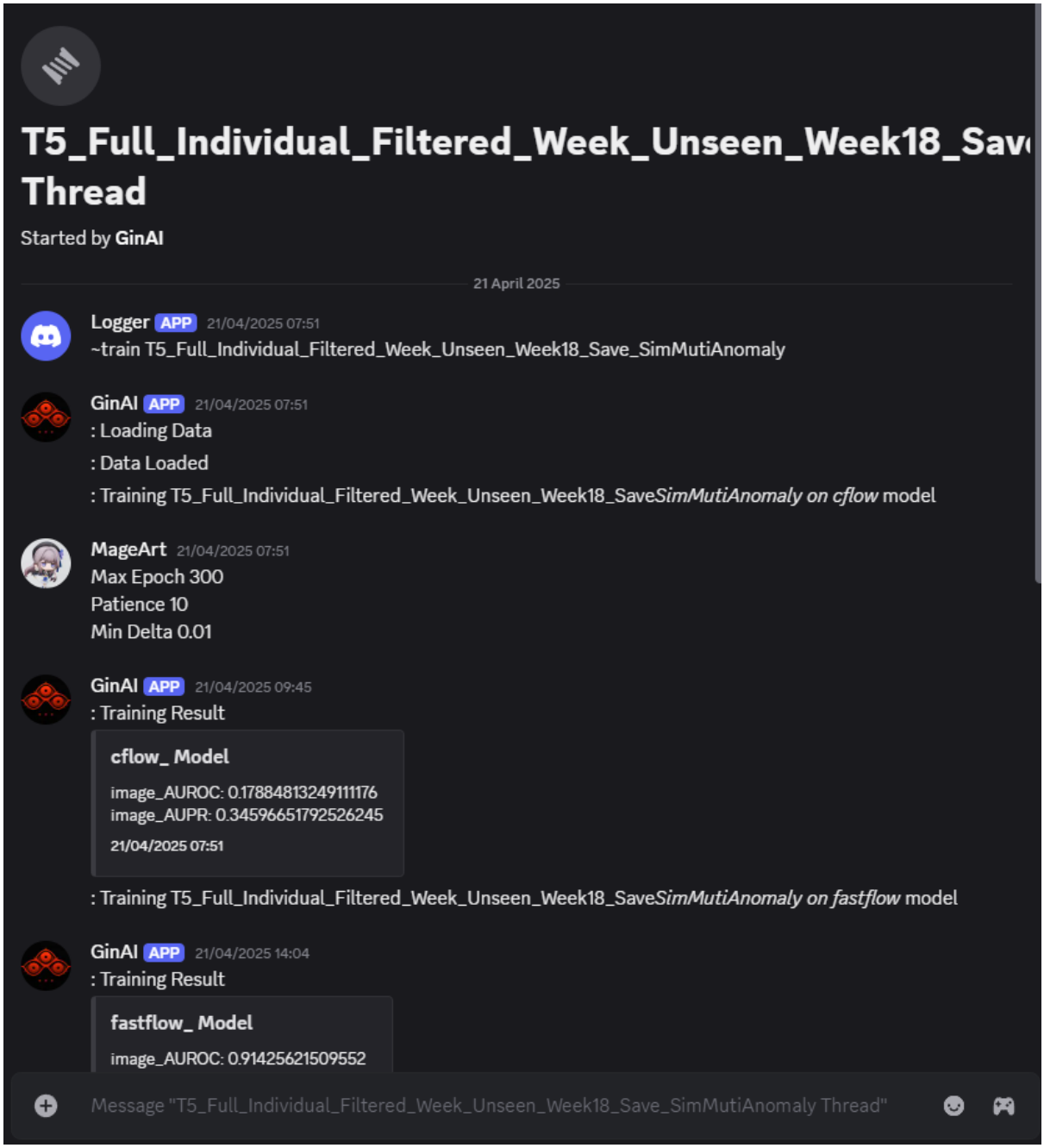


Figure 5.4.4: Dedicated Thread Logging for a full walkthrough of the training workflow

3. Help Command

There is a default ~help command on every channel to assist users by displaying the list of allowed commands for the specific channel. This assists users in knowing their current working context and prevents them from accidentally using commands restricted to other channels. The help message is constructed dynamically depending

on which webhook is being used and provides real-time guidance without recourse to external documentation.

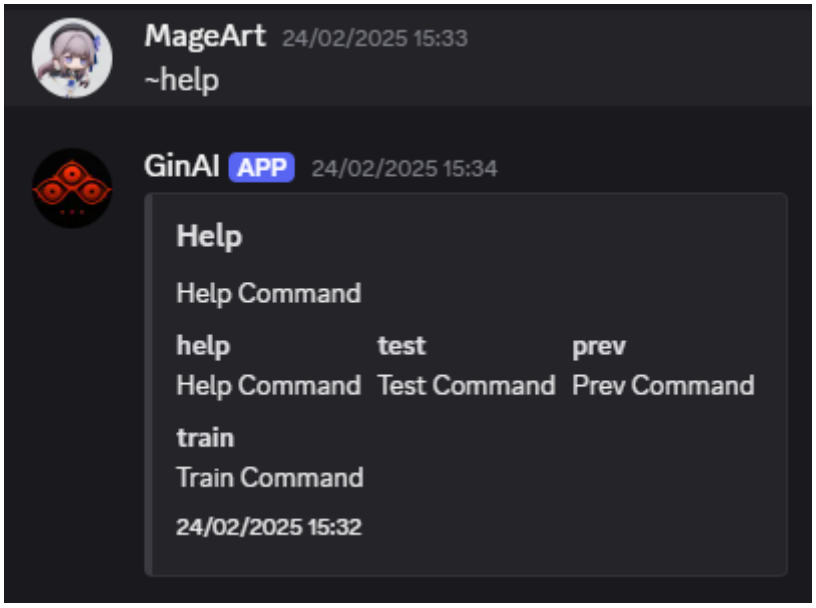


Figure 5.4.5: Help Command Output for a sample output in a Discord channel.

4. Prediction Workflow

Prediction has two main steps: setup and image submission. The user initially uses the setup command to select the trained model to use for prediction. Once the model is loaded and confirmed by the bot, the user then submits an image using the predict command. The model executes the image and returns the results along with a confidence level and classification output.

The prediction output is color-coded to indicate the severity or class of detection:

- Green – Normal: The plant or object is confidently normal, and no anomaly is detected.
- Blue – Potentially Normal: The sample is almost normal but close to the boundary and might need monitoring.
- Yellow – Potential Anomaly: There are minor irregularities that suggest the sample might be drifting towards an anomalous condition.
- Red – Anomaly Detected: There is a clear and high-confidence anomaly detected in the sample.

Each colour-coded response is presented with annotated images, prediction confidence scores, and links to the relevant thread for context or next steps.

Figure 5.4.6: Predict Setup Command

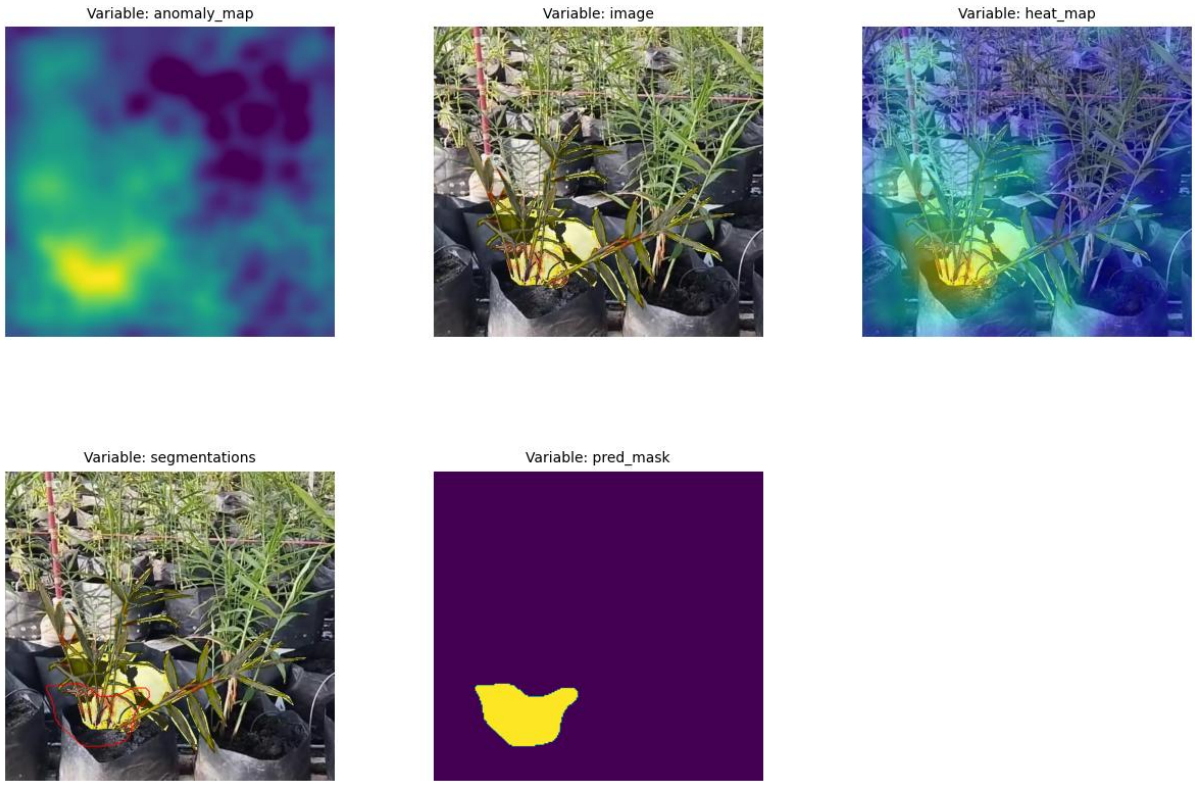
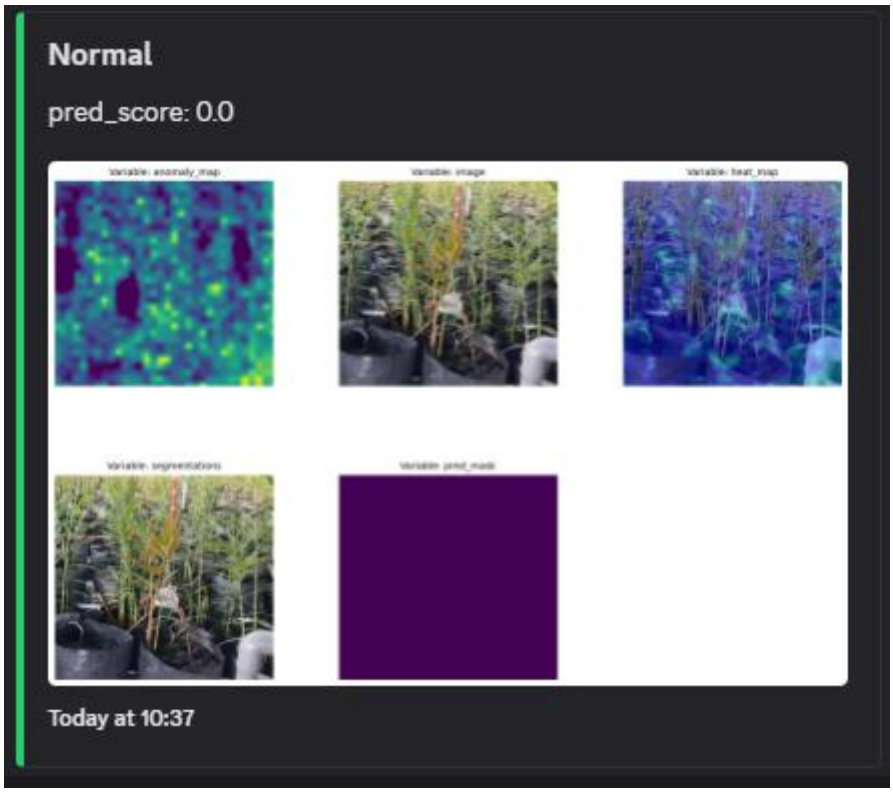
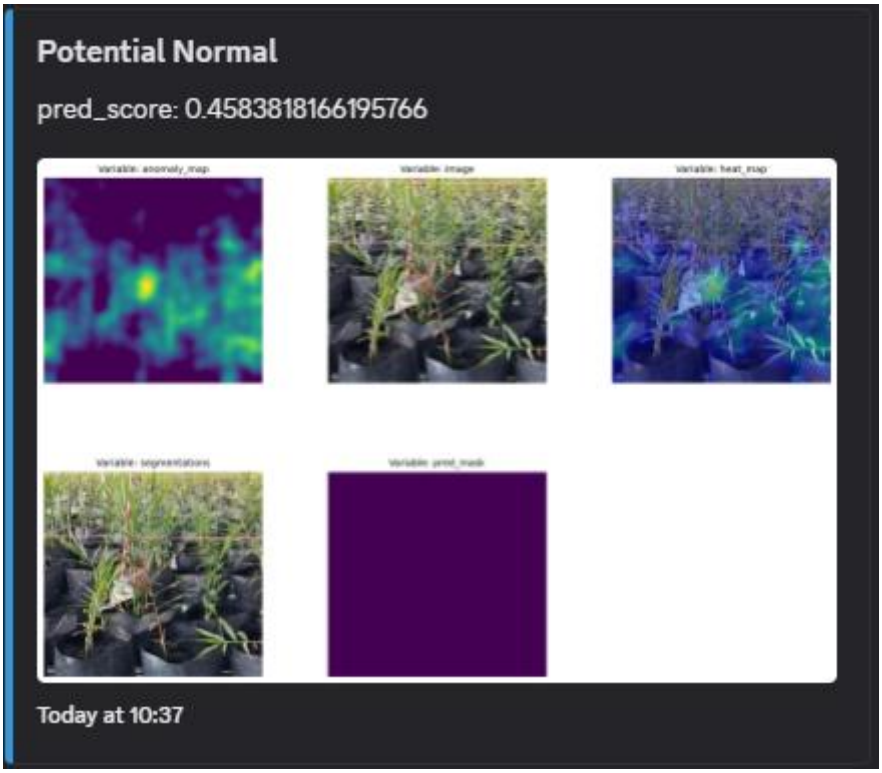


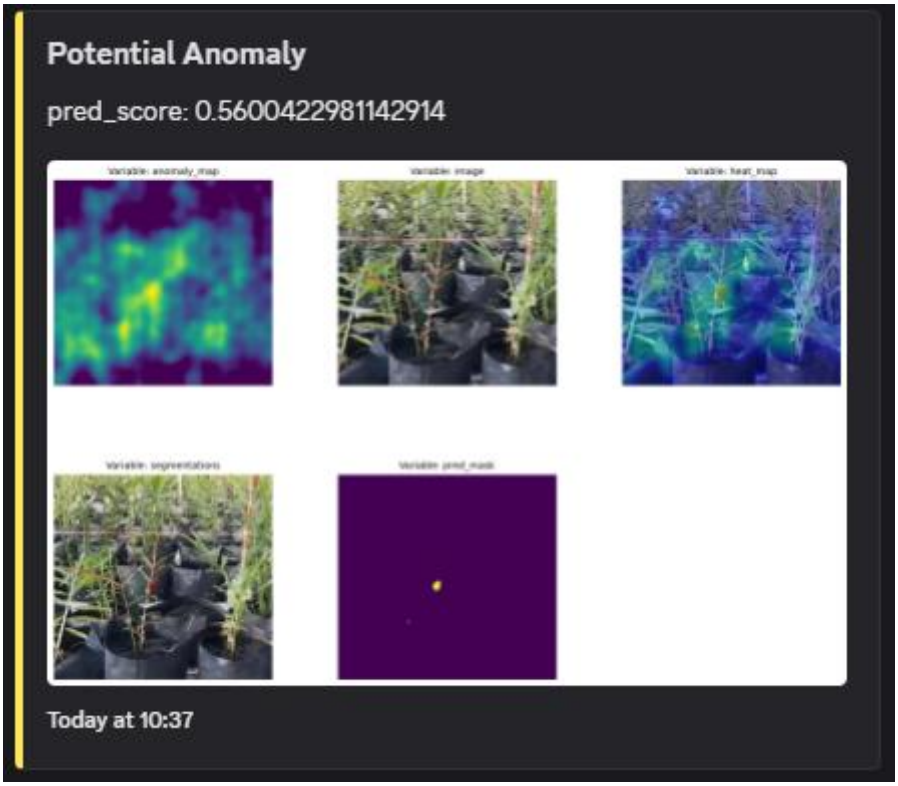
Figure 5.4.7: Predict Command Result



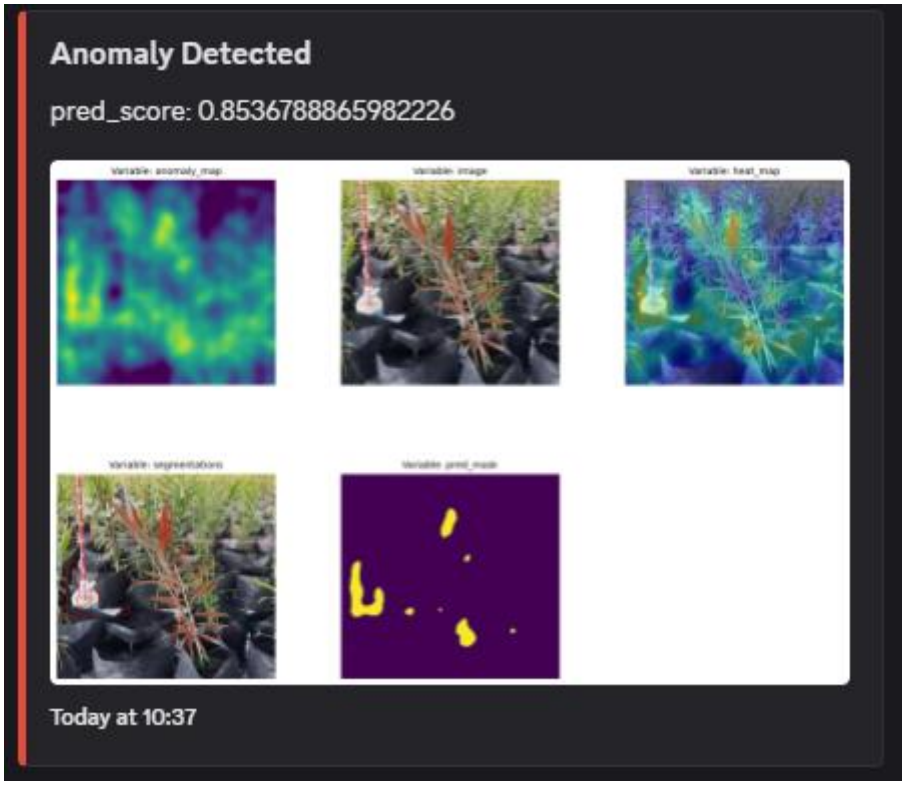
Figures 5.4.8 Visual examples of Green outputs



Figures 5.4.9 Visual examples of Blue outputs



Figures 5.4.10 Visual examples of Yellow outputs



Figures 5.4.11 Visual examples of Red outputs

5.5 Implementation Issues and Challenges

During the creation of the ginger plant anomaly detection system, several implementation problems and issues were brought about by the utilization of third-party software, shifting frameworks, and uncertainties regarding system design and scale. Among the most basic challenges was the result of utilizing Anomalib, an extremely efficient but quickly evolving library for anomaly detection. While Anomalib provides access to several cutting-edge models, the update frequency tends to also bring compatibility issues along. A couple of the models that previously were available are now deprecated or in production branches, and there is no possibility of using those in a production environment without an additional modification. Therefore, this project required a selective selection of stable versions and the building of an adaptive structure capable of accepting modification with minimal disruption. But in achieving this, it limited the range of successful anomaly detection models unless general rewrites of internal training and inferencing pipelines were performed.

Yet another major contributor was the input/output manipulation required to bridge the gap between Anomalib's rigid internal processes and the custom use cases needed by the system. Since Anomalib demands inputs to be provided in a specific format and inference be computed

in a format not easily applied to asynchronous environments like Discord or Flask APIs, custom wrappers and interceptors had to be constructed. These features enabled the system to feed dynamic inputs (e.g., images users upload) into the models and then record the output to transform it into a suitable form for display on Discord. This included not just common outputs like confidence scores but also annotated images and colour-coded interpretations for end users. These personalizations introduced layers of complexity, with the necessity of in-depth knowledge of Anomalib's codebase and frequent debugging with version updates.

The second principal development challenge was the uncertainty of the project's scope and outcome. As this was the developer's first experience of creating a production-grade system with real-time AI inference, Discord integration, and multi-threaded training pipelines, many design decisions had to be made on the fly. The system initially had a wide-brush stroke plan—such as providing anomaly detection through a Discord interface—yet no fixed map of how the individual modules (like model training, logging, user handling, and response prediction) were to be implemented or scaled up. This lack of clarity failed to enable prior planning for the system's size, memory requirements, or concurrency issues. It also posed the threat of architectural mismatches or incompatibilities down the road, especially if demand from users increases or the project is expanded to handle multiple simultaneous models or more intricate workflows.

Synchronizing multiple asynchronous systems—Discord bot, Flask API, and Anomalib training and prediction loops—was also difficult to align thread safety and performance. Discord and Flask are on separate event loops, and Anomalib's training procedures are blocking and CPU/GPU-bound by design. Ensuring that such features would be able to coexist and work together without going into interface lock-up mode or shared resource corruption needed extensive utilization of concurrency techniques and testing. This was exacerbated by the limited support and documentation available for using Anomalib and Discord.py simultaneously.

In general, the deployment of this system was a process of continuous adaptation and debugging. Despite the technical difficulties and the learning curve, these problems provided valuable lessons in software integration, real-time AI deployment, and future-proofing and modularity design. The end system, though imperfect and still in need of future maintenance,

is a working proof-of-concept that finds a balance between usability, model performance, and system robustness in an R&D environment.

5.6 Concluding Remark

The deployment of the Ginger Plant Anomaly Detection System has revealed the complexity and intricacy of integrating various technologies into a singular, working pipeline. In its essence, the system couples the strong strength of Anomalib for unsupervised anomaly detection with a Discord-based interaction model to offer a leading, accessible monitoring solution. Each component—from data preparation and training to inference and result sharing—was deliberately designed to work together in concert under a unified architecture, even with the inherent limitations of third-party dependencies and constantly shifting software libraries.

Through the utilization of a three-tier architecture (interface, processing, and data), the system compartmentalized its necessary functions to allow modular scalability and management. Flask delivered asynchronous processing successfully, Discord offered an interactive communication layer, and Python scripts served as a basis for workloads in anomaly detection. Webhook configurations, mappings of channel-threads, and tokenized environment variables were employed to address the issue of managing Discord commands on various channels and separation of concerns.

While the system in its current state functions as expected, the implementation process provided valuable lessons. It became clear that flexibility and anticipation are paramount, particularly when working with ever-evolving machine learning platforms like Anomalib. The system is designed with extensibility, and it is thus possible to include additional models, additional support datasets, and optimize interaction mechanisms in the future. Lastly, this deployment not only achieved the project objectives but also offered a scalable platform for future innovations in automated plant health monitoring.

Chapter 6

System Evaluation and Discussion

In this chapter, we will examine and evaluate the performance of the developed system by conducting thorough testing and examining the performance of the anomaly detection models implemented against their performance metrics. We will cover the testing environment, how we tested the performance of the system, and the obtained results from the testing phase. This chapter will also cover the issues faced while developing the system, particularly due to the dependencies on third-party software and the uncertainty in anomaly detection tasks. Additionally, we will analyze how far the project objectives were met and talk about the lessons learned while implementing. Finally, we will conclude by providing an insight into the overall success of the project and provide recommendations for any improvements and developments in the future.

6.1 System Testing and Performance Metrics

System testing and performance measurement are key aspects of ascertaining the effectiveness and correctness of an anomaly detection system. In this section, we provide an overview of the system's testing process, the performance metrics taken to analyze its performance, and how the system is tested for its robustness and correctness in detecting plant health anomalies. The testing process includes several key aspects:

- Dataset: The choice and creation of the dataset are significant elements in determining the range within which the system can operate at its best under varying conditions. The dataset should mirror real-world scenarios as much as possible, with variations in the health of plants and environmental factors.

- Training and Testing Configuration: The training configuration defines how the model is trained, outlining the specific configurations used to optimize its learning. This includes details such as the selection of the training set, test set, and if transformations or augmentations are made to the images.

- Selected Model: The choice of a model for anomaly detection influences the effectiveness with which it will detect deviations in the plant health data. The model should be capable of handling real-world data complexity and identifying subtle changes that could represent disease or stress in plants.

- Description of Metrics: There are various performance metrics for assessing the model's success at detecting anomalies. Typical metrics like accuracy, precision, recall, F1 score, ROC curves, and AUC provide information on how effectively the model identifies true positives (correct anomalies) and does not identify false positives or negatives.

- Visual Inspection: Visual inspection is an important part of system testing, especially in anomaly detection. By observing output images and visualized results, it is possible to interpret how well the model has localized anomalies and how the model responds to different types of inputs, i.e., normal, near-normal, and defective plant images.

Together, they provide a complete overview of the system's capability to detect plant anomalies, responsiveness to various factors, and overall performance in real-world applications. The following sections will discuss the dataset details, training setup, models chosen, performance metrics, and visual judgment results, providing an overall performance evaluation of the system.

6.1.1 Dataset

In the first phase of testing, significant efforts were made to organize and tune the dataset to enhance the model's capability to detect ginger plant anomalies. Dataset preparation involved several key phases, which were crucial for training and testing system performance. Following is the description of dataset preparation and the transformations conducted to determine its strength and validity of results:

1. **Dataset Composition**

   The dataset includes images taken over a period of four weeks with their respective week labels: Week 3, Week 8, Week 12, and Week 18. The images for each week are as follows:

- Week 3: 25 images
- Week 8: 65 images
- Week 12: 33 images
- Week 18: 107 images

These images were also divided into training and testing sets, with every third image in the sequence being selected for the test set.

2. **Preprocessing and Cropping**

   One of the initial problems observed was the non-uniform scaling of images, resulting in the loss of useful plant features and scale information. To bypass this, we cropped the images based on the largest region of interest (ROI) each week. This served to retain and normalize the most prominent characteristics of the plants in the dataset. By taking the largest ROI, we were able to minimize scale differences that could bewilder the model, particularly in detecting faint anomalies in plant health.

3. **Image Variations for Testing**

   In the test set, the following variations were added to give a comprehensive evaluation of the anomaly detection system:

   Good images:

   - **Contrast Adjustments**: The contrast of the good (normal) images was modified by adjusting the contrast factor. The contrast was modified with factors of 0.7, 0.9, 1.1, and 1.3 to simulate minor environmental changes that can naturally occur in a ginger farm.
   -

   Bad (defective) images:

   - **Hue Shift**: Hue shift was utilized to simulate disease or stress. The shift was applied down 15% and 30% and rotated the hue by 180 degrees, inducing artificial anomalies in the plant colour.
   - **Dying Variations**: One of the significant challenges in plant disease detection is handling subtle signs of plant deterioration. To simulate this, three levels of

hue, saturation, and value (HSV) alterations were created to produce dying plant variations. The alterations were defined as follows:

a) Variation 1: (10, 50, 50) - Slight alterations in hue, saturation, and value.
b) Variation 2: (20, 70, 70) - Moderate alterations in hue, saturation, and value.
c) Variation 3: (30, 90, 90) - Larger changes, simulating more severe symptoms of plant decay.

Using the formulae:

```python
brown_h = np.clip(h - hue_adj, 10, 30)
brown_s = np.clip(s - sat_adj, 50, 255)
brown_v = np.clip(v - val_adj, 50, 255)
hsv_brown = cv2.merge([brown_h, brown_s, brown_v])
```

Figure 6.1.1.1 Formulae of Dying Variation

This modification created a wide range of abnormal-looking plants, closely resembling real degradation due to factors such as nutrient deficiency, disease, or age.

4. Splitting the Dataset

a) Train and Test Set Split: Images were divided into train and test sets, with a crucial test set choosing approach:
   - They chose every third image from the available images for the test set.
   - This aided in getting a good representation of images from every week so that there would be no bias from overfitting the model to certain periods.

b) Evaluation Set: To guarantee that the performance of the model was always assessed in various circumstances, half of the test set was also assigned as the evaluation set. This assisted in creating more realistic settings for model validation, such that the model could be validated on data it had not seen before.

5. Images Use for Training

It only used normal images to train it so that the model wasn't influenced by any labelled anomalies during learning. This also enabled the model to learn healthy plant patterns first before it was exposed to anomalous conditions.

By contrast adjustments and introducing very few changes to the regular images, the model was rendered more robust to identify even slight changes in the healthy plants, which would later allow it to identify anomalies more precisely.

6. **Model Testing and Anomaly Detection**

   The divided and preprocessed dataset was also used to validate the anomaly detection model by applying the transformations on the normal (good) images and incorporating some other types of anomalies (bad) to represent real-case scenarios. This enabled the model to detect subtle and gross changes in plant health and functionality.

   The contrast, hue shift, and dying variations provided a diverse range of test cases to assess the model's ability to handle diverse forms of plant anomalies.

This refined dataset and its careful preparation served to ensure the system's ability to successfully identify anomalies under varying conditions. The issues of scaling, transformations of images, and dataset splitting were duly addressed, allowing for a more stable and accurate evaluation of the model's performance in anomaly detection in plant health.

6.1.2 Training and Testing Configuration

Having prepared the dataset, the next most critical task in building a successful anomaly detection system is establishing a good training and testing regime. This phase is necessary to ensure that the models not only learn useful patterns in the data but also generalize well to new, unseen situations without overfitting. The settings chosen in this project are informed by the demands of the Anomalib framework, real-world hardware constraints, and experimental objectives of plant anomaly detection using unsupervised learning. This section presents training and testing settings, such as training parameters, resource utilization, and model persistence strategy.

Training Configuration

1. Training Strategy and Early Stopping:
    - A loss step scheduler is used in the training process to monitor the model's training loss over epochs.

    - Early stopping with patience of 10 epochs is employed to prevent overtraining and save computation resources, whereby training will stop if no improvement is seen after 10 consecutive validation steps.

    - Mode is 'min', trying to minimize the loss function during training, and minimum delta is 0.01, which defines the minimum improvement in loss to be considered an improvement.

    - Verbose mode is enabled to display real-time diagnostics and progress during training, tracking the learning behaviour and detecting problems early.

2. Thresholding Mechanism and Evaluation Task:
    - Instead of a fixed predefined threshold, the system uses F1 Adaptive Thresholding. It automatically selects the best threshold according to the F1 score, which is a trade-off between precision and recall and is particularly useful in scenarios of imbalanced data or blurry anomaly boundaries.

    - The task is defined as 'classification', casting the anomaly detection as a binary problem normal or anomalous which aligns with the format of test data and simplifies evaluation.

    - Performance metrics are AUROC (Area Under the Receiver Operating Characteristic Curve) and AUPR (Area Under the Precision-Recall Curve). These metrics are suitable for evaluating anomaly detection models since they are not affected by class imbalance and provide a general view of performance across all thresholds.

3. Hardware and Runtime Settings:

- Maximum epochs is 300, providing ample training time while balancing against the risk of overfitting. We choose this number since most of the model did not reach the maximum epochs.

- Device auto-selection (device=auto) and accelerator auto-setting (accelerator=auto) are used to automatically choose the most appropriate available computing hardware (GPU or CPU) for computing based on compatibility and performance regardless of the environment of the system.

- Due to hardware limitations of available hardware, i.e., CPU and memory size, data loading workers are restricted to 2. This reduces parallel fetching of data but prevents system crashes and allows smoother runtime during training.

4. Train/Test Split Strategy:
   - In this setup, a manual train/test split is enforced by defaulting the split ratio to 0, providing full control over dataset distribution.

   - This option guarantees there is no training and test data leakage between datasets, preserving the integrity of the test process and ensuring realistic test conditions.

5. Image Preprocessing and Format:
   - All the images are resized to 256 × 256 pixels, which is a common input size for the majority of convolutional models. Standardization also conserves GPU memory usage and compatibility with all the models realized in Anomalib.

   - The images are converted to RGB colour format, which is what the majority of trained vision backbones expect and which preserves colour information critical to anomaly detection, such as hue and saturation-based variation in plant health.

   - At the end of the training, all models are saved in PyTorch (.pt) format to facilitate compact storage and easy reloading for inference or further fine-tuning. This also facilitates future portability and integration into other systems.

Testing Configuration

- Testing is minimalist and realistic, replicating the way the system would be utilized in the real world in deployment scenarios.

- Test images are shown in the raw form, and there is no preprocessing or transformation done at test time. This enables the same variations, defects, and environmental conditions that appear during real-time operation to appear during testing.

- This straightforward test setup achieves two goals: first, it provides a clear measure of the model's capacity to handle raw input; second, it does not introduce any artificial gains that would distort the performance metrics.

Overall, this system's train/test configuration is aimed at establishing a controlled yet realistic environment for model performance evaluation. By sacrificing rigour (in the form of meticulous parameter tuning and data handling) for ease (especially in testing), this configuration offers a good foundation for meaningful performance evaluation and reliable field deployment. It also models real-world constraints such as limited computational resources and evolving software tools, which makes the system not just effective but also sustainable.

6.1.3 Chosen Model

The selection of anomaly detection models is a key determinant of the overall system's accuracy, robustness, and practical applicability. Five unsupervised anomaly detection models from the Anomalib library were chosen for this project based on their strengths, architectural novelty, and demonstrated performance in earlier research and preliminary experiments. Each of the models was chosen to play a distinct role in detecting subtle to obvious anomalies under diverse conditions and test weeks of the ginger plant dataset. Their combination enables exhaustive evaluation and improves the capability of the system to detect a wide variety of plant health issues.

The five models that have been selected to be evaluated are:

- CFlow

  CFlow was chosen because of its strong performance on difficult plant image datasets with diverse structures and lighting conditions. Its conditional normalizing flow model

enables it to learn fine-grained spatial feature representations, making it especially helpful in identifying differences in healthy and unhealthy plant structures in big datasets. CFlow showed consistently stable results in preliminary experiments, particularly in high-resolution plant shape and texture change extraction.

- FastFlow

  FastFlow is included because of its speed and real-time inference capability, together with its stable anomaly detection under different visual conditions. While it does not necessarily produce the most accurate heat maps, its overall anomaly classification is good and consistent. FastFlow is a nice speed versus accuracy balance model that would be suitable for systems requiring quick feedback, i.e., real-time monitoring systems.

- PatchCore

  PatchCore is selected due to its stability and versatility across a variety of datasets. It is a patch-wise feature embedding approach with a memory bank, making it very effective at general-purpose anomaly detection. PatchCore performs well even when the fine details are not as salient, and it had a strong resilience to false positives in initial testing. It is often a "baseline strong performer," making it a valuable component of comparative studies and deployment scenarios.

- Reverse Distillation

  This model is included primarily for its prospects in detecting anomalies such as shadows, wilting, or fallen-over plants — conditions that are typically more challenging for normal models. Reverse Distillation relies on a student-teacher framework, where the disagreement between the outputs of the teacher and student represents potential anomalies. It is strong in detecting scene-level inconsistencies and subtle changes in plant posture or environment, which was effective in week 12 and week 18 testing when plants were collapsing or dying.

- STFPM

  STFPM was selected due to its high performance in fine localization of anomalies. It takes advantage of multi-scale feature matching and can identify very slight deviations from the learned normal distribution. During visual inspection, STFPM consistently

produced high-resolution anomaly maps well correlated with areas of interest in diseased plants. Its accuracy and resolution place it at the top of the models for individuals interested in fine anomaly localization.

Through the use of this heterogeneous collection of models, the system benefits from the unique strengths that each one brings. Whereas some of the models, like STFPM and CFlow, are directed towards pixel-wise high-detail localization, others like PatchCore and FastFlow offer more general anomaly detection with shorter response times. Reverse Distillation completes the others by filling in structural and environmental anomalies. Together, these models offer a broad toolkit for plant health analysis and anomaly detection in a variety of real-world agricultural contexts.

6.1.4 Explanation of Metrics

To most effectively measure the performance of the anomaly detection models, a diverse set of metrics was used to ensure both the classification accuracy and ability to generalize to new, unseen data were properly investigated. The metrics provide insight into the model's decision-making, particularly when the datasets are imbalanced and the anomaly severity levels. Manual evaluation was carried out by comparing each prediction output image-by-image against the ground truth. Note that all the evaluation images were completely unseen to the models during training so that there could be an unbiased and fair model generalization test.

The following key metrics were utilized:

- True Negative (TN): Number of correctly identified normal (healthy) plant images.
- False Positive (FP): Number of normal images incorrectly identified as anomalies.
- False Negative (FN): Number of actual anomalous images misclassified as normal.
- True Positive (TP): Number of correctly identified anomalous images.

From these values, a variety of derived metrics were calculated:

- Accuracy: Overall correctness of the model:

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

- Precision: Proportion of true anomalies among all detected anomalies:

$$Precision = \frac{TP}{TP + FP}$$

- Recall (Sensitivity): Ability to detect actual anomalies:

$$Recall = \frac{TP}{TP + FN}$$

- Specificity: Ability to correctly identify normal samples:

$$Specificity = \frac{TN}{TN + FP}$$

- F1 Score: Harmonic mean of precision and recall, balancing both false positives and false negatives:

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

- False Positive Rate (FPR): Likelihood of misclassifying a normal image as an anomaly:

$$FPR = \frac{FP}{TN + FP}$$

- False Negative Rate (FNR): Likelihood of missing an actual anomaly:

$$FNR = \frac{FN}{TP + FN}$$

- Balanced Accuracy: Average of sensitivity and specificity, useful when classes are imbalanced:

$$Balanced\ Accuracy = \frac{Sensitivity + Specificity}{2}$$

- Youden's Index (J): A metric that summarizes the performance by maximizing the distance from random chance classification:

$$J = Sensitivity + Specificity - 1$$

- Negative Predictive Value (NPV): Probability that a negative prediction truly corresponds to a normal image:

$$NPV = \frac{TN}{TN + FN}$$

These metrics were vital in interpreting model performance outside binary classification, delivering insight into over-predicting anomaly models of anomalies, failing to detect weak instances, and predicting confidence to believe within the true agricultural setting.

Besides manually vetted metrics, two automatically produced metrics from the training phase were utilized as well:

- AUROC (Area Under the Receiver Operating Characteristic Curve): Assesses the discriminative ability of the model in separating normal and anomalous samples at varying thresholds. The higher AUROC indicates better discrimination.

- AUPR (Area Under the Precision-Recall Curve): Most helpful with imbalanced datasets, examining precision-recall trade-offs.

The values for AUROC and AUPR were recorded during training and indicate performance on the test split utilized by the training setup. However because these are developed during training from artificial variations or static test sets, they serve only as secondary pointers and not as the final source of assessment. The underlying performance analysis was calculated from manual verification and confusion matrix-based metrics to better emulate the real-world challenges and model reliability in actual deployments.

6.1.5 Visual Evaluation

In addition to quantitative metrics, visual evaluation also played a critical role in establishing the real-world performance of every anomaly detection model. This involved using original cropped-size images every week, allowing the models to predict over a greater number of pixels and ensuring the unambiguous representation of plant features. Visual inspection was imperative not just to validate prediction effectiveness but also in determining spatial location of detected faults—something non-scalar measurements cannot quantify.

For realistic "good" image evaluation, a sample was considered correctly predicted if an anomaly was not detected in the plant zone. When anomalies were detected by the model in background areas (e.g., soil, pots, or shadows outside of the plant zone), predictions were not

regarded as false positives. This prevented the assessment from disproportionately focusing on the system's ability to detect plant health compared to unrelated image noise.

For "bad" image evaluation, predictions were only considered valid anomalies if the predicted area overlapped most of the actual anomaly on the plant, say areas of discolouration, distortion, or wilt. If anomaly predictions were outside the region of interest (ROI) or indicated healthy plant tissue, they were excluded from performance metrics. This stringent ROI-based validation ensured that model predictability was specifically linked with a proper semantic understanding of plant health conditions, not simply pixel-level faults.

With the aid of this visual validation process, evaluation gave precedence to interpretability and applicability in the real world in a field setup, ensuring models were not just mathematically sound but visually and semantically understandable when utilized for ginger plant health surveillance.

6.2 Result

This section gives the comprehensive performance testing results of the system's anomaly detection on five selected models — CFlow, FastFlow, PatchCore, Reverse Distillation, and STFPM — on four diverse weekly datasets labelled as Week 3, Week 8, Week 12, and Week 18. Each week's dataset represents ginger plants' cumulative growth and fluctuation in health obtained under true field conditions. These data sets are chosen specifically to contain a broad range of plant health states and environment states, with the test conditions consequently being realistic and difficult. All images used in the evaluation phase here are novel; i.e., they were not present in training at all and therefore provide an effective basis through which to test true generalization ability.

Each week's data was subjected to four feature-based test conditions: Normal images, Contrast Down variations, Hue Downshifts, and Dying Plant simulations. These test cases reflect typical plant health degradation situations, such as loss of leaf colour, reduced light exposure, and starting wilting. The aim was to find out the models' ability to effectively pick out anomalies in both subtle and obvious plant stress conditions. For all these differences, confusion matrices and a full set of performance metrics were computed to provide both statistical and practical information regarding each model's performance under stress and normal conditions.

Performance metrics include commonly used classification metrics such as Accuracy, Precision, Recall (Sensitivity), Specificity, F1 Score, False Positive Rate, False Negative Rate, Balanced Accuracy, Youden's Index, and Negative Predictive Value. These values were acquired by manually verifying each model's prediction output against ground-truth labels by pixel-based visual anomaly inspection. Precautions were taken to distinguish between true detection and false alarms by removing predictions that were outside the plant region of interest (ROI), particularly when anomalies were being wrongly detected in the background.

Besides the manually verified metrics, the models also produced automated performance metrics such as AUROC (Area Under Receiver Operating Characteristic Curve) and AUPR (Area Under Precision-Recall Curve) while training. These are other metrics of the models' threshold optimization and classification capability while performing the task of anomaly detection. Although AUROC and AUPR are useful high-level summaries, manually validated metrics provide ground-truth confirmation of whether or not the predictions made were indeed useful and reliable in field conditions.

In visual validation, all test images were kept at their original crop size to supply optimal pixel area for anomaly prediction. This allowed a more detailed assessment of how well the model could localize anomalies and how reliably it could identify prominent patterns. Anomaly detections were only taken as true positives when these were occurring within the ROI of anomalous plants. Predictions on healthy plants or background noise were not being counted to avoid unfair inflation of false positive or false negative rates. Healthy image variations like changes, in contrast, were only included as good detections if no anomalies were being falsely detected.

The subsequent subsections provide a comprehensive breakdown of results for each week's data. Each test condition is then preceded by confusion matrix visualization, a table of performance metrics, and a brief interpretation of model strengths and weaknesses in the condition. This comprehensive method presents a clear and measurable appreciation of the system's actual performance and highlights which models and conditions are most responsible for successful or failed detection.

6.2.1 Weeks Evaluation Results

Week 3

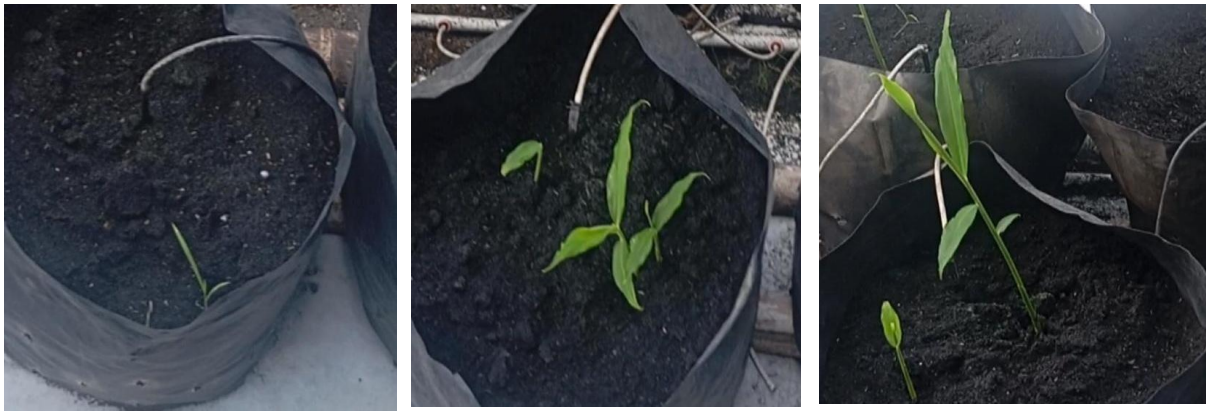
1. Overview

   Week 3 assessment is focused on a relatively early stage of ginger plant growth. Various augmentation images were used for testing each depicting healthy or artificially unhealthy plant conditions under various visual augmentations. The conditions are nicely chosen to evaluate the anomaly detection capability of the selected models, which were trained on normal plant images alone. The Week 3 dataset is comprised of smaller and thinner-developed plants, which may have unique challenges to proper anomaly detection by offering fewer visual cues and thinner vegetative cover.

2. Test Conditions and Augmentations

   Four test scenarios were prepared for Week 3:

   - Normal images: Unmodified crop-sized top-down captures of healthy plants.



Figures 6.2.1.1 Week 3 Normal-Present-Smalles, Medium, Largest Plant Images

   - Contrast Up and Down: Images modified using contrast adjustments (0.7, 0.9, 1.1 and 1.3 factors) to simulate environmental lighting issues.

Figures 6.2.1.2 Week 3 Contrast Adjustment 0.7, 0.9, 1.1 and 1.3

- Hue Down: Images where hue values were shifted down (15% and 30% of 180) to simulate changes in leaf coloration due to nutrient issues or early disease symptoms.

Figures 6.2.1.3 Week 3 Hue Down 15 and 30.

- Dying Variation: Images processed with HSV transformations simulating leaf browning and drying using predefined hue, saturation, and value reduction combinations.

Figures 6.2.1.4 Week 3 Dying Variation 1, 2 and 3

Each of these categories was intended to validate how the models respond not only to genuine anomalies but also to borderline or ambiguous visual cues.

3. Confusion Matrix

   For each test case, model predictions were manually validated against expected outcomes, and confusion matrices were constructed accordingly. This included detailed counts of:

The result as shown below:

Good	Cflow			fastflow			patchcore			reverse distillation			stfpm		
	Correct	Total	Accuracy	Correct	Total	Accuracy	Correct	Total	Accuracy	Correct	Total	Accuracy	Correct	Total	Accuracy
Normal	5	6	83.33%	6	6	100.00%	6	6	100.00%	5	6	83.33%	6	6	100.00%
contrast down 0.7	3	4	75.00%	4	4	100.00%	4	4	100.00%	4	4	100.00%	3	4	75.00%
contrast down 0.9	3	3	100.00%	3	3	100.00%	3	3	100.00%	3	3	100.00%	3	3	100.00%
contrast up 1.1	3	4	75.00%	4	4	100.00%	4	4	100.00%	3	4	75.00%	4	4	100.00%
contrast up 1.3	2	3	66.67%	3	3	100.00%	3	3	100.00%	2	3	66.67%	3	3	100.00%
bad															
dying 1	4	4	100.00%	4	4	100.00%	3	4	75.00%	3	4	75.00%	2	4	50.00%
dying 2	0	2	0.00%	2	2	100.00%	2	2	100.00%	0	2	0.00%	2	2	100.00%
dying 3	1	3	33.33%	2	3	66.67%	3	3	100.00%	2	3	66.67%	3	3	100.00%
hue down 15	1	6	16.67%	0	6	0.00%	0	6	0.00%	2	6	33.33%	0	6	0.00%
hue down 30	1	5	20.00%	1	5	20.00%	1	5	20.00%	2	6	33.33%	5	6	83.33%

Table 6.2.1.1 Week 3 Result

Each confusion matrix provides insight into the model's sensitivity and its robustness against visual artifacts that resemble real plant issues.

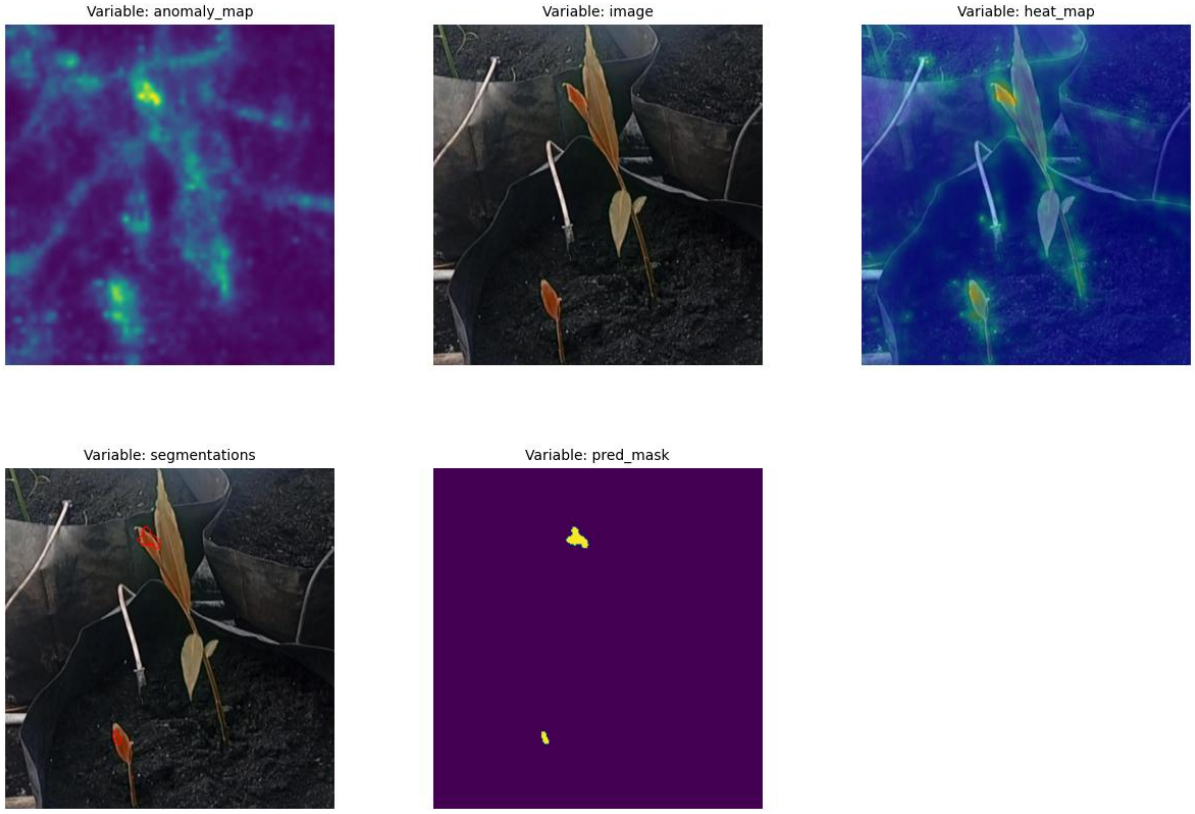


Figure 6.2.1.5 Week 3 Result STFPM

a. CFlow

	Predicted Positive	Predicted Negative
Actual Positive	7	13
Actual Negative	4	16

Table 6.2.1.2 Week 3 CFlow Confusion Matrix

b. Fastflow

	Predicted Positive	Predicted Negative
Actual Positive	9	11
Actual Negative	0	20

Table 6.2.1.3 Week 3 Fastflow Confusion Matrix

c. PatchCore

	Predicted Positive	Predicted Negative
Actual Positive	9	11
Actual Negative	0	20

Table 6.2.1.4 Week 3 PatchCore Confusion Matrix

d. Reverse Distillation

	Predicted Positive	Predicted Negative
Actual Positive	9	12
Actual Negative	3	17

Table 6.2.1.5 Week 3 Reverse Distillation Confusion Matrix

e. STFPM

	Predicted Positive	Predicted Negative
Actual Positive	12	9
Actual Negative	1	19

Table 6.2.1.6 Week 3 STFPM Confusion Matrix

4. Performance Metrics

Based on the confusion matrix, the following metrics were computed:

model	cflow	fastflow	patchcore	reverse distillation	stfpm
Accuracy	57.50%	72.50%	72.50%	63.41%	75.61%
Precision	63.64%	100.00%	100.00%	75.00%	92.31%
Recall (Sensitivity)	35.00%	45.00%	45.00%	42.86%	57.14%
Specificity	80.00%	100.00%	100.00%	85.00%	95.00%
F1 Score	45.16%	62.07%	62.07%	54.55%	70.59%
False Positive Rate	20.00%	0.00%	0.00%	15.00%	5.00%
False Negative Rate	65.00%	55.00%	55.00%	57.14%	42.86%
Balanced Accuracy	57.50%	72.50%	72.50%	63.93%	76.07%
Youden's Index (J)	15.00%	45.00%	45.00%	27.86%	52.14%
Negative Predictive Value	55.17%	64.52%	64.52%	58.62%	67.86%
AUROC	81.00%	68.50%	79.25%	76.75%	75.75%
AUPR	82.69%	72.28%	83.09%	76.45%	81.79%

Table 6.2.1.7 Week 3 Performance Metrics

These metrics were critical in identifying not just accuracy, but also the trade-offs between catching all anomalies and avoiding false alarms. Additionally, AUROC and AUPR scores were extracted from the training phase and included as baseline model quality indicators.

5. Visual Evaluation Insight

   The Week 3 visual evaluation step was insightful regarding the response of each model to normal and anomalous visual cues, especially when viewed at full crop resolution. Anomaly predictions were overlaid on the test images in this step, and qualitative observations were recorded on how well the predictions aligned with the actual plant conditions.

   - CFlow showed consistent alignment with normal samples, often producing clean prediction masks with little or no false positives. Its behaviour on anomaly images—particularly under hue shift and dying leaf simulations—was difficult to interpret. In many cases, the model neither indicated anomalies with confidence nor showed strong heatmap hints. This is worrying regarding whether it is responding appropriately or just not registering more subtle changes.

- FastFlow performed adequately on good and bad cases. On normal samples and contrast changes, it maintained acceptable prediction clarity. On bad samples, the model had some ability to highlight suspicious areas, but sensitivity was low. Visual cues like browning or colour changes were not strongly suggested, reflecting a lack of discriminative power for mild anomaly transformations.

- PatchCore also followed the same trend as FastFlow, with good performance on good images and tolerably medium performance on bad ones. On dying variation and hue-down cases, it could weakly indicate areas of the problem but was not able to cross the anomaly threshold. Like FastFlow, it appeared to sense something was wrong but was not confident enough to explicitly call it an anomaly.

- Reverse Distillation varied slightly in detecting early discolouration and yellowing even in "good" images, which, though it points to sensitivity, can lead to false positives on borderline cases. In anomalous cases, especially with dying variants, the model was capable of detecting affected regions but could be refined with further tuning to amplify anomaly signal strength. Its performance is suggestive of a strong underlying ability, but with a threshold that must be tuned.

- STFPM was the most stable in terms of visual feedback. It accurately differentiated normal and abnormal samples and consistently emphasized affected areas under hue-down and dying conditions. However, its performance on hue-down 15% variation cases was somewhat of a concern. It tended to label minor hue changes as anomalies even when these were within acceptable visual limits for healthy plants. Nonetheless, among all models, STFPM had the best localization and confidence on visual anomaly maps.

Of all the models, hue-down changes were the hardest to evaluate. These sorts of changes consistently produced anomaly flags on actual "good" as well as "bad" samples, hiding the model's ability to identify subtle changes in coloration. This points

out the need for either more complex hue-sensitive augmentation processing or refined thresholding logic.

Overall, the Week 3 visual evaluation accentuates both the model strengths and weaknesses present. It confirms the necessity for continued human-in-the-loop validation, especially where model outputs are responsive to uncertain colour or contrast variation in real-world plant images.

The Week 3 visual evaluation indicates that while all models possessed a baseline ability to delineate normal from anomalous plant conditions, they exhibited variable sensitivity and accuracy. CFlow performed well under normal samples and struggled to make definitive outputs on anomalous ones, which called its reliability under subtle disturbances into question. FastFlow and PatchCore were largely consistent but lacked the depth to pick up on less obvious signs of stress such as early colour shifts. Reverse Distillation yielded good discolouration and shadow-based anomaly detection with high potential, though, in some good samples, it was bordering on being too sensitive. STFPM was most consistent, with well-localized and sharp detections and good trade-off between sensitivity and specificity. Hue-down transformations, however, exposed a blind spot common to all the models, which tended to confuse even visually healthy images. These results highlight the importance of tailored preprocessing and postprocessing methods, particularly for ambiguous or colour-based anomaly cues in plant health monitoring.

Week 8

1. Overview

   In Week 8, the dataset increased significantly in terms of size and visual variety, posing a more taxing test of each model's capacity for generalization. Week 8 introduced additional diverse leaf patterns, lighting conditions, and anomaly severities, including slight yellowing, moderate leaf curling, and increased image complexity. The test focused on the strength of the model under this more challenging distribution, namely their performance on both normal and augmented good samples (e.g., contrast variations) and the more subtle bad samples, such as slight hue variations and early signs of dying. The visual results allow more subtle observations about model strengths and weaknesses under field-like conditions, where anomaly signals are practically

never binary and may lie in visually ambiguous ranges. The Week 8 results are a critical waypoint for dividing those models that can scale effectively as data complexity grows and those that require further tuning or domain-specific adaptation.

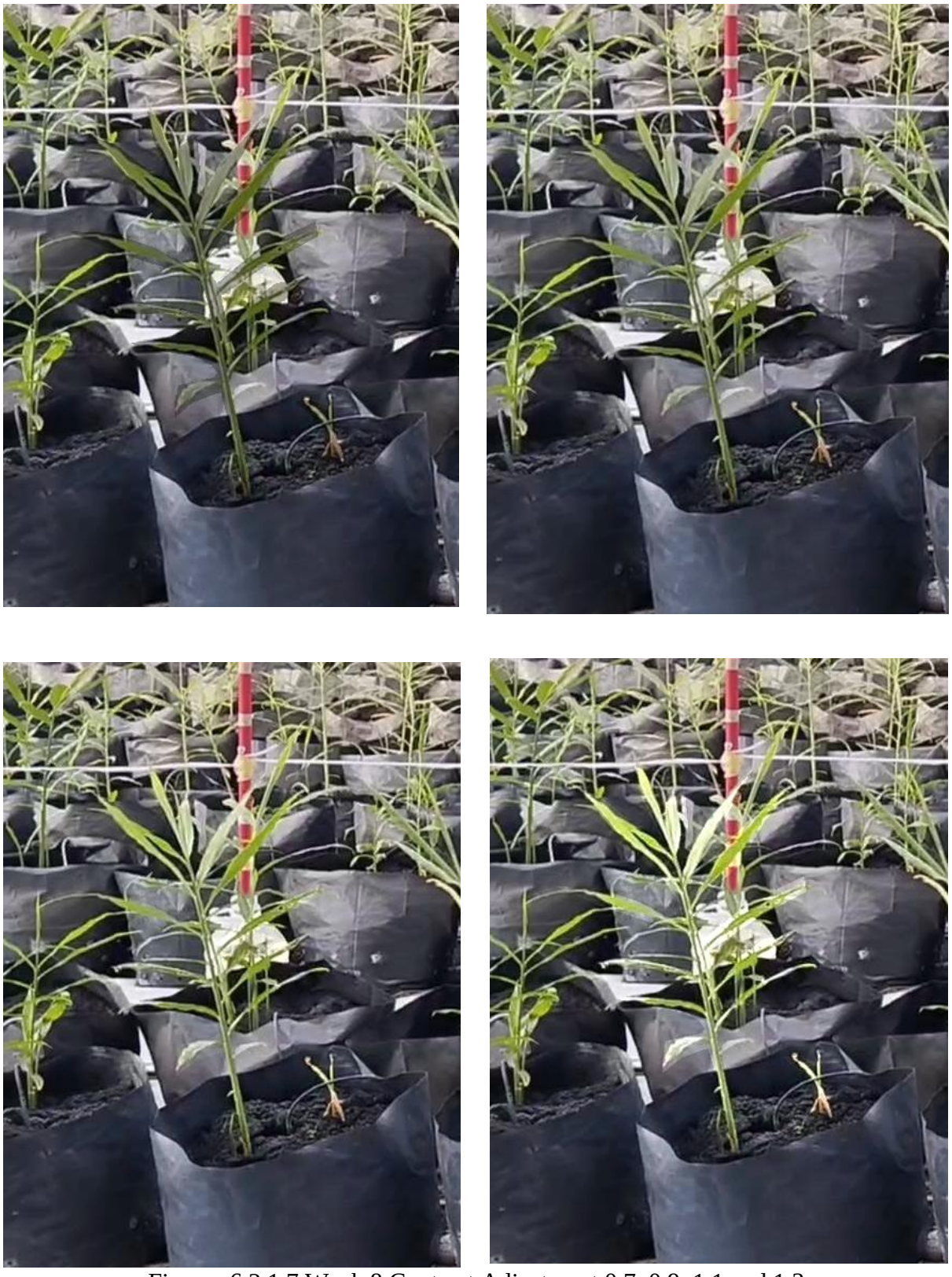
2. Test Conditions and Augmentations

   Four test scenarios were prepared for Week 8:

   - Normal images: Unmodified crop-sized top-down captures of healthy plants.

Figures 6.2.1.6 Week 8 Normal-Present-Smalles, Medium, Largest Plant Images

- Contrast Up and Down: Images modified using contrast adjustments (0.7, 0.9, 1.1 and 1.3 factors) to simulate environmental lighting issues.



Figures 6.2.1.7 Week 8 Contrast Adjustment 0.7, 0.9, 1.1 and 1.3

- Hue Down: Images where hue values were shifted down (15% and 30% of 180) to simulate changes in leaf coloration due to nutrient issues or early disease symptoms.

Figures 6.2.1.8 Week 8 Hue Down 15 and 30.

- Dying Variation: Images processed with HSV transformations simulating leaf browning and drying using predefined hue, saturation, and value reduction combinations.

Figures 6.2.1.9 Week 8 Dying Variation 1, 2 and 3

Each of these categories was intended to validate how the models respond not only to genuine anomalies but also to borderline or ambiguous visual cues.

3. Confusion Matrix

For each test case, model predictions were manually validated against expected outcomes, and confusion matrices were constructed accordingly. This included detailed counts of:

The result as shown below:

Good	Cflow			fastflow			patchcore			reverse distillation			stfpm		
	Correct	Total	Accuracy	Correct	Total	Accuracy	Correct	Total	Accuracy	Correct	Total	Accuracy	Correct	Total	Accuracy
Normal	12	12	100.00%	11	12	91.67%	12	12	100.00%	9	9	100.00%	9	9	100.00%
contrast down 0.7	11	14	78.57%	14	14	100.00%	14	14	100.00%	9	9	100.00%	9	9	100.00%
contrast down 0.9	11	11	100.00%	11	11	100.00%	11	11	100.00%	11	11	100.00%	11	11	100.00%
contrast up 1.1	9	11	81.82%	10	11	90.91%	11	11	100.00%	11	11	100.00%	11	11	100.00%
contrast up 1.3	5	5	100.00%	4	5	80.00%	5	5	100.00%	12	13	92.31%	14	14	100.00%
bad															
dying 1	3	11	27.27%	8	11	72.73%	9	11	81.82%	7	8	87.50%	6	8	75.00%
dying 2	3	10	30.00%	10	10	100.00%	10	10	100.00%	11	12	91.67%	11	12	91.67%
dying 3	1	9	11.11%	8	9	88.89%	9	9	100.00%	9	9	100.00%	9	9	100.00%
hue down 15	4	13	30.77%	5	13	38.46%	2	13	15.38%	2	11	18.18%	5	11	45.45%
hue down 30	2	10	20.00%	0	10	0.00%	3	10	30.00%	5	13	38.46%	9	13	69.23%

Table 6.2.1.8 Week 8 Result

Each confusion matrix provides insight into the model's sensitivity and its robustness against visual artifacts that resemble real plant issues.

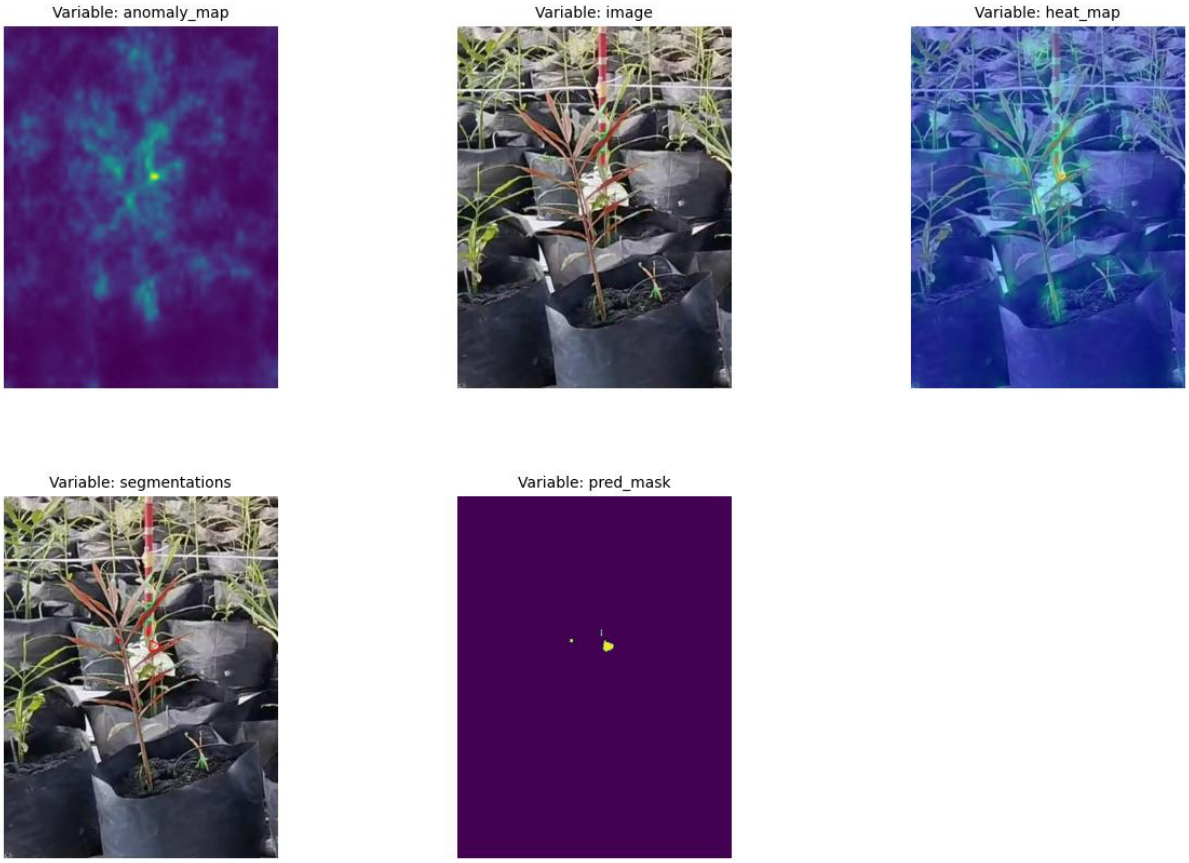


Figure 6.2.1.10 Week 8 Result STFPM

a. CFlow

	Predicted Positive	Predicted Negative
Actual Positive	13	13
Actual Negative	5	16

Table 6.2.1.9 Week 8 CFlow Confusion Matrix

b. Fastflow

	Predicted Positive	Predicted Negative
Actual Positive	31	22
Actual Negative	3	50

Table 6.2.1.10 Week 8 Fastflow Confusion Matrix

c. PatchCore

	Predicted Positive	Predicted Negative
Actual Positive	33	20
Actual Negative	0	53

Table 6.2.1.11 Week 8 PatchCore Confusion Matrix

d. Reverse Distillation

	Predicted Positive	Predicted Negative
Actual Positive	34	19
Actual Negative	1	52

Table 6.2.1.12 Week 8 Reverse Distillation Confusion Matrix

e. STFPM

	Predicted Positive	Predicted Negative
Actual Positive	40	13
Actual Negative	0	54

Table 6.2.1.13 Week 8 STFPM Confusion Matrix

4. Performance Metrics

Based on the confusion matrix, the following metrics were computed:

model	cflow	fastflow	patchcore	reverse distillation	stfpm
Accuracy	57.55%	76.42%	81.13%	81.13%	87.85%
Precision	72.22%	91.18%	100.00%	97.14%	100.00%
Recall (Sensitivity)	24.53%	58.49%	62.26%	64.15%	75.47%
Specificity	90.57%	94.34%	100.00%	98.11%	100.00%
F1 Score	36.62%	71.26%	76.74%	77.27%	86.02%
False Positive Rate	9.43%	5.66%	0.00%	1.89%	0.00%
False Negative Rate	75.47%	41.51%	37.74%	35.85%	24.53%
Balanced Accuracy	57.55%	76.42%	81.13%	81.13%	87.74%
Youden's Index (J)	15.09%	52.83%	62.26%	62.26%	75.47%
Negative Predictive Value	54.55%	69.44%	72.60%	73.24%	80.60%
AUROC	75.99%	76.68%	84.12%	85.62%	80.69%
AUPR	84.75%	75.99%	89.04%	88.83%	77.17%

Table 6.2.1.14 Week 8 Performance Metrics

These metrics were critical in identifying not just accuracy, but also the trade-offs between catching all anomalies and avoiding false alarms. Additionally, AUROC and AUPR scores were extracted from the training phase and included as baseline model quality indicators.

5. Visual Evaluation Insight

   Week 8 visual evaluation provided deeper insight into how effective each model had been in coping with increasing dataset complexity and higher plant condition variability. With increasing use of full crop resolution for testing, models were assessed on the basis of their ability to accurately localize anomalies and avoid false detections, particularly as increasingly variable contrast and hue changes were included.

   - CFlow performed well on normal samples, often generating clean masks without mistakenly labeling the background. However, its performance in bad samples was still inconsistent. The model did not conclusively react to many visually apparent anomalies, especially under slight hue variations or progressive plant rot. Though its low sensitivity might restrict false positives, it also raised questions about its general detection confidence and reliability.

   - FastFlow did better robustness on normal samples, perhaps due to better background handling and fine-tuning in prediction confidence. Its anomaly detection on bad images was slightly improved, with clearer heatmaps around affected areas. Still, subtle changes such as early-stage decay or slight hue downshifts continued to be underrepresented. The model appeared to need stronger anomaly triggers to be confident about classifying an image as abnormal.

   - PatchCore was especially strong on handling normal as well as anomaly images. It performed excellent discrimination on normal examples without the need for explicit background filtering. Under anomaly instances particularly with dwindling variations the model produced logical heatmap activations. While it failed to cross over anomaly thresholds sometimes on more sublet cases, its

localization remained consistent with actual areas of interest, hence serving as a reliable middle-ground candidate.

- Reverse Distillation further improved its interpretability by showing abnormal regions clearly even in complex plant geometries. It handled dying and hue-shifted variants more firmly than it had in Week 3 and managed to differentiate between normal versus suspect plant condition. Its sensitivity also managed to detect incipient discoloration and minute structural weakness. However, it still required slight calibration to reduce false positives from benign hue changes.

- STFPM nonetheless performed better than all the rest of the models when it came to map distinctness and anomaly localization. It highly discriminated normal from poor samples, often yielding intricate and specific heatmaps even for very minor changes. Hue-down cases were well taken care of, though at the 15% threshold some of the visual clues were borderline leading to slight changes at times being classified wrongly as anomalies. Nonetheless, STFPM still performed better with minimal misclassification and good spatial accuracy.

Hue-down 15% was similarly a recurring problem on all models. This change was subtle enough that it was within normal limits but was able to mislead some models into marking them as anomalies. This merely serves to highlight the importance of better anomaly thresholding logic and possibly re-thinking if such subtle hue changes would even be considered anomalies under field conditions.

In short, Week 8 visual checks reminded that while all models have improved predictions, particularly for clear cases, minor plant health changes remain difficult to address without advanced tuning or domain-knowledgeful boost. STFPM, PatchCore, and Reverse Distillation proved most helpful, but all models exhibited features that must be investigated further when determining future deployment strategy.

Week 8's visual evaluation confirms continued improvement in model stability and detection confidence, with almost all models demonstrating greater capability in responding to more sophisticated plant structure and denser leaf pattern. CFlow continued its excellence in precise

predictions of typical samples but remained limited in decisiveness when subjected to subtle anomalies its indecision in the ability to discriminate minor visual impairments remained an ongoing shortcoming. FastFlow also fared marginally better in Week 3 with improved attention and reduced noise in predictions, yet still lacked the depth for routine flagging of borderline anomalies, especially color-related changes. PatchCore also achieved a fair performance with virtues in especially "dying" changes where indication was strongly prominent, but did exhibit some conservatism towards considering gentle deviations as anomalies. Reverse Distillation demonstrated good visual acuteness, correctly identifying discolored and shadowed regions, though its heightened sensitivity occasionally picked up non-critical differences, indicating that thresholding could be improved. STFPM once more outperformed the others, demonstrating excellent localization accuracy and robust anomaly detection—even in visually ambiguous cases. However, hue-down 15% conversions remained a problematic case for all but efficiently still managed to miss some anomalies or detect them falsely in healthy samples. These results emphasize the importance of high color sensitivity calibration and validate STFPM's leadership as a prevailing model for accurate plant anomaly detection at this level of development.

Week 12

1. Overview

   The Week 12 analysis is based on a more mature stage of ginger plant development with denser foliage and more organized spatial plant patterning compared to earlier stages. This week's data set is a variety of test images, normal and synthetically transformed samples that simulate usual signs of stress such as discoloration, contrast variation, and wilting leaves. Controlled augmentations have generated these test samples that mimic real-world environmental variation and physiological imperfections.

   The larger plant size and density in Week 12 provides a more informative set of visual features, enabling the models to better recognize abnormal patterns. However, this introduces the added complexity of distracting background features and more complex spatial relationships between healthy and infected areas. All the anomaly detection models being compared are still trained using normal plant images only to maintain the purity of the unsupervised learning framework. The Week 12 exam is a mid-point

benchmark, gauging how far the models generalize to more visually advanced and biologically developed plant states without first seeing anomalous cases.

2. Test Conditions and Augmentations

   Four test scenarios were prepared for Week 12:

   - Normal images: Unmodified crop-sized top-down captures of healthy plants.

Figures 6.2.1.11 Week 12 Normal-Present-Smalles, Medium, Largest Plant Images

   - Contrast Up and Down: Images modified using contrast adjustments (0.7, 0.9, 1.1 and 1.3 factors) to simulate environmental lighting issues.

Figures 6.2.1.12 Week 12 Contrast Adjustment 0.7, 0.9, 1.1 and 1.3

- Hue Down: Images where hue values were shifted down (15% and 30% of 180) to simulate changes in leaf coloration due to nutrient issues or early disease symptoms.

Figures 6.2.1.13 Week 12 Hue Down 15 and 30.

- Dying Variation: Images processed with HSV transformations simulating leaf browning and drying using predefined hue, saturation, and value reduction combinations.

Figures 6.2.1.14 Week 12 Dying Variation 1, 2 and 3

Each of these categories was intended to validate how the models respond not only to genuine anomalies but also to borderline or ambiguous visual cues.

3. Confusion Matrix

   For each test case, model predictions were manually validated against expected outcomes, and confusion matrices were constructed accordingly. This included detailed counts of:

   The result as shown below:

Good	Cflow			fastflow			patchcore			reverse distillation			stfpm		
	Correct	Total	Accuracy	Correct	Total	Accuracy	Correct	Total	Accuracy	Correct	Total	Accuracy	Correct	Total	Accuracy
Normal	3	5	60.00%	3	5	60.00%	5	5	100.00%	8	9	88.89%	9	9	100.00%
contrast down 0.7	5	7	71.43%	6	7	85.71%	7	7	100.00%	3	4	75.00%	4	4	100.00%
contrast down 0.9	4	5	80.00%	4	5	80.00%	5	5	100.00%	5	5	100.00%	5	5	100.00%
contrast up 1.1	5	6	83.33%	5	6	83.33%	6	6	100.00%	4	6	66.67%	6	6	100.00%
contrast up 1.3	4	5	80.00%	4	5	80.00%	5	5	100.00%	4	4	100.00%	4	4	100.00%
bad															
dying 1	1	5	20.00%	3	5	60.00%	4	5	80.00%	5	7	71.43%	6	7	85.71%
dying 2	1	7	14.29%	4	7	57.14%	7	7	100.00%	5	5	100.00%	5	5	100.00%
dying 3	1	3	33.33%	3	3	100.00%	3	3	100.00%	7	7	100.00%	7	7	100.00%
hue down 15	1	5	20.00%	2	5	40.00%	2	5	40.00%	2	5	40.00%	2	5	40.00%
hue down 30	2	8	25.00%	3	8	37.50%	2	8	25.00%	3	4	75.00%	3	4	75.00%

Table 6.2.1.15 Week 12 Result

Each confusion matrix provides insight into the model's sensitivity and its robustness against visual artifacts that resemble real plant issues.

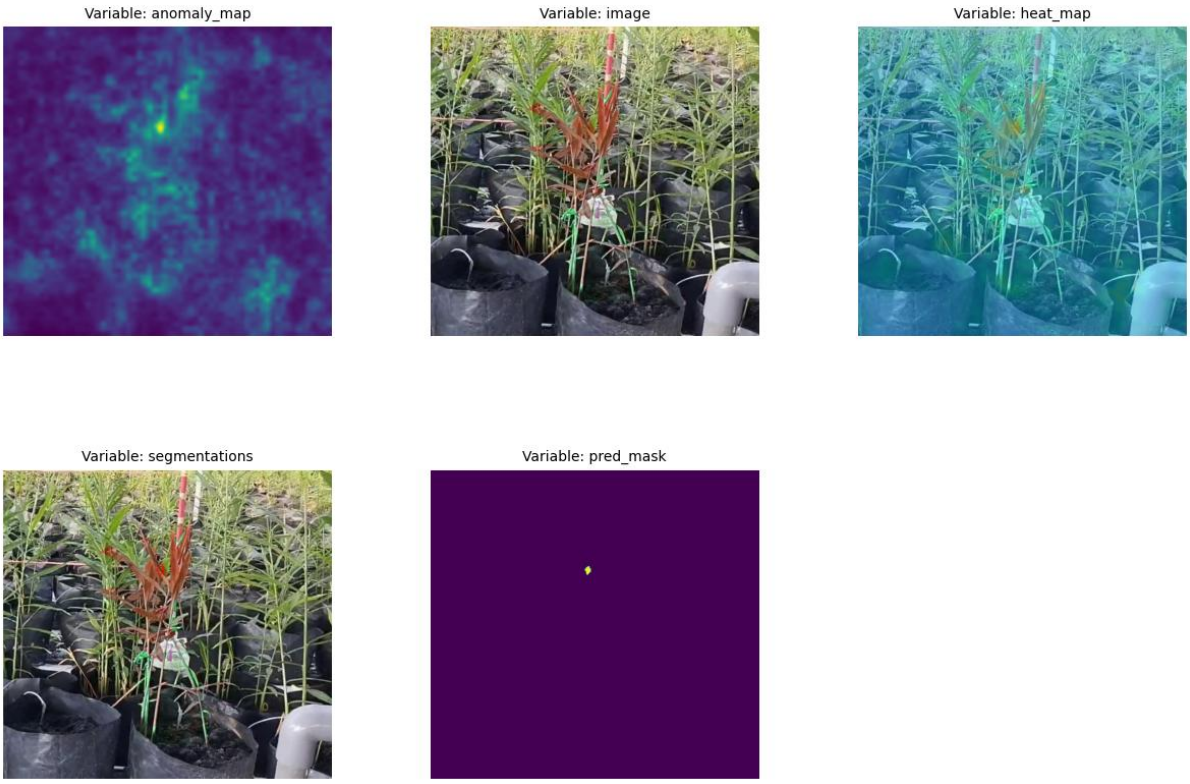


Figure 6.2.1.15 Week 12 Result STFPM

a. CFlow

	Predicted Positive	Predicted Negative
Actual Positive	6	13
Actual Negative	7	16

Table 6.2.1.16 Week 12 CFlow Confusion Matrix

b. Fastflow

	Predicted Positive	Predicted Negative
Actual Positive	15	13
Actual Negative	6	22

Table 6.2.1.17 Week 12 Fastflow Confusion Matrix

c. PatchCore

	Predicted Positive	Predicted Negative
Actual Positive	18	10

Actual Negative	0	28

Table 6.2.1.18 Week 12 PatchCore Confusion Matrix

d. Reverse Distillation

	Predicted Positive	Predicted Negative
Actual Positive	22	6
Actual Negative	4	24

Table 6.2.1.19 Week 12 Reverse Distillation Confusion Matrix

e. STFPM

	Predicted Positive	Predicted Negative
Actual Positive	23	5
Actual Negative	0	28

Table 6.2.1.20 Week 12 STFPM Confusion Matrix

4. Performance Metrics

Based on the confusion matrix, the following metrics were computed:

model	cflow	fastflow	patchcore	reverse distillation	stfpm
Accuracy	48.21%	66.07%	82.14%	82.14%	91.07%
Precision	46.15%	71.43%	100.00%	84.62%	100.00%
Recall (Sensitivity)	21.43%	53.57%	64.29%	78.57%	82.14%
Specificity	75.00%	78.57%	100.00%	85.71%	100.00%
F1 Score	29.27%	61.22%	78.26%	81.48%	90.20%
False Positive Rate	25.00%	21.43%	0.00%	14.29%	0.00%
False Negative Rate	78.57%	46.43%	35.71%	21.43%	17.86%
Balanced Accuracy	48.21%	66.07%	82.14%	82.14%	91.07%
Youden's Index (J)	-3.57%	32.14%	64.29%	64.29%	82.14%
Negative Predictive Value	48.84%	62.86%	73.68%	80.00%	84.85%
AUROC	7.14%	78.83%	75.77%	73.09%	76.59%
AUPR	75.00%	78.49%	70.91%	71.60%	66.63%

Table 6.2.1.21 Week 12 Performance Metrics

These metrics were critical in identifying not just accuracy, but also the trade-offs between catching all anomalies and avoiding false alarms. Additionally, AUROC and

AUPR scores were extracted from the training phase and included as baseline model quality indicators.

5. Visual Evaluation Insight

   The Week 12 visual inspection step provides a mid-stage insight into how each anomaly detection model responded to ginger plants with moderate growth. The dataset for this week has varied images, which is a well-balanced mix of normal and abnormal conditions, along with simulated augmentations such as dying leaves, contrast changes, and hue shifts. The vegetation at this growth stage is denser and more developed than in Week 8, with greater shadow and texture complexity, offering subtle challenges to anomaly detection.

   - CFlow provided a decent but sub-par performance on normal images. Its predictions were occasionally interrupted by background elements, and prediction masks were noisy. On anomalous images, especially in the dying or hue-down case, CFlow performed exceedingly poorly, producing indistinct or useless heatmaps. The model did not highlight critical areas numerous times or discern between minor visual noise vs. actual anomalies, an indication of insufficient sensitivity at this mid-stage growth.

   - FastFlow mimicked CFlow's performance with normal predictions moderately well aligned but disrupted by visual noise, possibly plant shadow or minor textural differences. On the anomaly samples, FastFlow was unstable and less convincing with faint heatmap signals on obvious cases like dying leaves. Its inability to flag anomalies confidently or localize them sharply reduces its practical utility without further tuning or refinement.

   - PatchCore, in contrast, performed exceptionally well, particularly on normal samples where its emphasis on small details made it impervious to background noise. Rather astonishingly, PatchCore detected plant shadows as anomalies, which, although technically incorrect, speaks volumes about its hypersensitivity visually. On dying samples and colour change, the model exhibited clear and confident detection, outperforming both CFlow and FastFlow by a long way.

- Reverse Distillation provided solid and stable performance in most test cases. On good images, it maintained low anomaly scores, even though background interference impacted its consistency to some extent. On bad samples, especially those with dying plant symptoms, the model possessed decent capability to highlight anomalous regions, but its responses lacked the consistency of PatchCore or STFPM. It was equivocal on hue-down at 15%, but at 30% it began to respond more emphatically, suggesting that its sensitivity curve is more commensurate with stronger visual changes.

- STFPM continued to be a leading performer. It maintained perfect alignment on normal samples with no false positives and exhibited excellent precision in identifying dying and highly altered plants. In hue-down tests, STFPM behaved as intended it had negligible detection at 15% shifts but became effective at 30%, which means it has a well-calibrated sensitivity threshold. Among all models, STFPM offered the best localization accuracy and anomaly confidence combination.

Across the board, hue-down changes remained difficult, particularly at the 15% level where changes were too subtle for some models to mark as anomalies with any certainty. Models like CFlow and FastFlow would miss these changes, while PatchCore and STFPM responded more reliably at larger hue changes. Visual clutter caused by background noise continued to be a challenge, particularly for CFlow and Reverse Distillation.

In conclusion, Week 12 confirmed the differences in the way each model handles mid-stage plant imagery. Whereas PatchCore and STFPM featured high sensitivity and interpretability, models like CFlow and FastFlow lagged, with unclear anomaly signals and noisy predictions. This stage demonstrated that as plant size and complexity increase, model robustness to background artifacts and weak cues becomes ever more crucial making STFPM and PatchCore the leading candidates for field deployment in practice.

The Week 12 visual examination demonstrates an improved delineation between normal and abnormal plant conditions, with all models exhibiting varying degrees of consistency and finesse. CFlow performed modestly on good samples but lacked consistency, with a tendency to yield noisy or incomplete heatmaps on anomalous inputs. FastFlow exhibited similar performance reasonably good for healthy plants but struggled with subtle anomalies, which resulted in ambiguous predictions. PatchCore was good at precise anomaly localization, especially on the dying leaf samples, but tended to mistake shadows for abnormal as it was very sensitive. Reverse Distillation performed very well on both good and defective images, though being moderately troubled by background artifacts, meaning that it required improved context filtering. STFPM yielded the most consistent and visually accurate results, with correct identification of normal and anomalous regions and minimal false positives. Hue-down transformations nonetheless remained difficult for all models, particularly at lower intensity shifts, reflecting a continued limitation for colour-based anomaly detection. These findings highlight again the need for calibrated detection thresholds and more advanced handling of subtle colour gradations to enhance reliability for plant health monitoring.

Week 18

1. Overview

   Week 18 assessment is the most advanced stage of ginger plant growth documented in this study and contains the biggest collection of images so far. Unlike in the earlier weeks, the visual scene in Week 18 is compactly occupied by mature ginger plants placed next to one another, eliminating visible background elements. This offers a unique test environment wherein anomaly detection models must attend only to fine-grained visual details of the plants themselves, decoupled from contextual background separation.

   Anomaly localization is harder but perhaps more accurate without non-plant areas and high plant density, assuming the model can successfully discern subtle intra-plant anomalies. The test images include original healthy samples and augmented versions mimicking real-world issues like leaf discolouration, colour variations, and contrast degradation. The vast amount of images in this set provides a solid foundation for evaluating model scalability, stability, and sensitivity under conditions highly akin to full-field agricultural applications.

Because of the homogeneity of the scene and the maturity of the plants, Week 18 is a critical checkpoint for the performance of anomaly detection models on dense, complex, and semantically homogeneous data. The performance here suggests the viability of a model being put into production environments where false positives have higher costs and accuracy in highlighting plant-level anomalies matters.

2. Test Conditions and Augmentations

   Four test scenarios were prepared for Week 18:

   - Normal images: Unmodified crop-sized top-down captures of healthy plants.

Figures 6.2.1.16 Week 18 Normal-Present-Smalles, Medium, Largest Plant Images

   - Contrast Up and Down: Images modified using contrast adjustments (0.7, 0.9, 1.1 and 1.3 factors) to simulate environmental lighting issues.

Figures 6.2.1.17 Week 18 Contrast Adjustment 0.7, 0.9, 1.1 and 1.3

- Hue Down: Images where hue values were shifted down (15% and 30% of 180) to simulate changes in leaf coloration due to nutrient issues or early disease symptoms.


Figures 6.2.1.18 Week 18 Hue Down 15 and 30.

- Dying Variation: Images processed with HSV transformations simulating leaf browning and drying using predefined hue, saturation, and value reduction combinations.



Figures 6.2.1.19 Week 18 Dying Variation 1, 2 and 3

Each of these categories was intended to validate how the models respond not only to genuine anomalies but also to borderline or ambiguous visual cues.

3. Confusion Matrix

For each test case, model predictions were manually validated against expected outcomes, and confusion matrices were constructed accordingly. This included detailed counts of:

The result as shown below:

Good	Cflow			fastflow			patchcore			reverse distillation			stfpm		
	Correct	Total	Accuracy	Correct	Total	Accuracy	Correct	Total	Accuracy	Correct	Total	Accuracy	Correct	Total	Accuracy
Normal	9	16	56.25%	16	16	100.00%	16	16	100.00%	16	16	100.00%	16	16	100.00%
contrast down 0.7	7	18	38.89%	16	18	88.89%	18	18	100.00%	17	18	94.44%	18	18	100.00%
contrast down 0.9	5	15	33.33%	13	15	86.67%	15	15	100.00%	13	15	86.67%	15	15	100.00%
contrast up 1.1	9	19	47.37%	17	19	89.47%	19	19	100.00%	19	19	100.00%	19	19	100.00%
contrast up 1.3	8	20	40.00%	17	20	85.00%	20	20	100.00%	18	20	90.00%	20	20	100.00%
bad															
dying 1	9	21	42.86%	10	21	47.62%	20	21	95.24%	17	21	80.95%	18	21	85.71%
dying 2	5	11	45.45%	9	11	81.82%	11	11	100.00%	11	11	100.00%	10	11	90.91%
dying 3	11	20	55.00%	19	20	95.00%	20	20	100.00%	20	20	100.00%	20	20	100.00%
hue down 15	2	16	12.50%	3	16	18.75%	3	16	18.75%	4	16	25.00%	5	16	31.25%
hue down 30	7	20	35.00%	19	20	95.00%	19	20	95.00%	12	20	60.00%	18	20	90.00%

Table 6.2.1.22 Week 18 Result

Each confusion matrix provides insight into the model's sensitivity and its robustness against visual artifacts that resemble real plant issues.

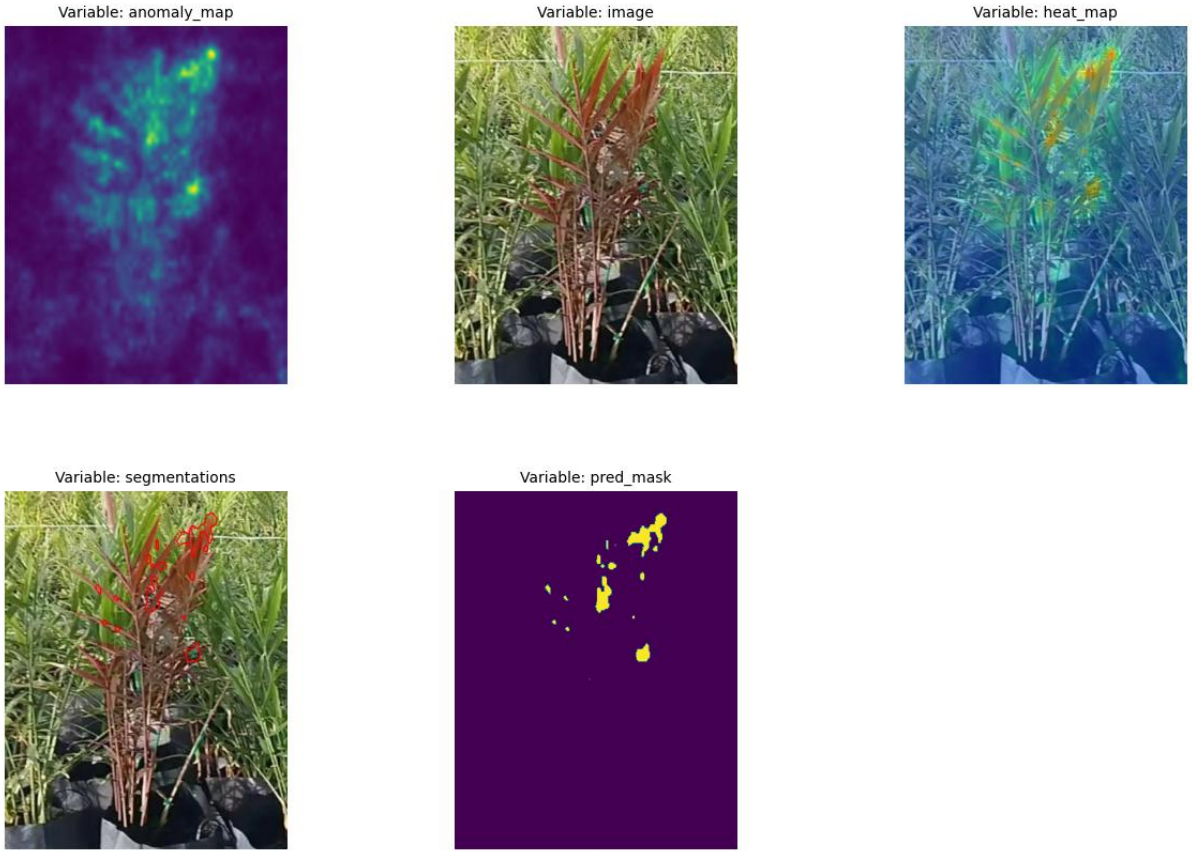


Figure 6.2.1.20 Week 18 Result STFPM

a. CFlow

	Predicted Positive	Predicted Negative
Actual Positive	34	13
Actual Negative	50	16

Table 6.2.1.23 Week 18 CFlow Confusion Matrix

b. Fastflow

	Predicted Positive	Predicted Negative
Actual Positive	60	28
Actual Negative	9	79

Table 6.2.1.24 Week 18 Fastflow Confusion Matrix

c. PatchCore

	Predicted Positive	Predicted Negative
Actual Positive	73	15
Actual Negative	0	88

Table 6.2.1.25 Week 18 PatchCore Confusion Matrix

d. Reverse Distillation

	Predicted Positive	Predicted Negative
Actual Positive	64	24
Actual Negative	5	83

Table 6.2.1.26 Week 18 Reverse Distillation Confusion Matrix

e. STFPM

	Predicted Positive	Predicted Negative
Actual Positive	71	17
Actual Negative	0	88

Table 6.2.1.27 Week 18 STFPM Confusion Matrix

4. Performance Metrics

Based on the confusion matrix, the following metrics were computed:

model	cflow	fastflow	patchcore	reverse distillation	stfpm
Accuracy	40.91%	78.98%	91.48%	83.52%	90.34%
Precision	40.48%	86.96%	100.00%	92.75%	100.00%
Recall (Sensitivity)	38.64%	68.18%	82.95%	72.73%	80.68%
Specificity	43.18%	89.77%	100.00%	94.32%	100.00%
F1 Score	39.53%	76.43%	90.68%	81.53%	89.31%
False Positive Rate	56.82%	10.23%	0.00%	5.68%	0.00%
False Negative Rate	61.36%	31.82%	17.05%	27.27%	19.32%
Balanced Accuracy	40.91%	78.98%	91.48%	83.52%	90.34%
Youden's Index (J)	-18.18%	57.95%	82.95%	67.05%	80.68%
Negative Predictive Value	41.30%	73.83%	85.44%	77.57%	83.81%
AUROC	17.88%	91.43%	91.86%	88.78%	97.48%
AUPR	34.60%	93.03%	94.11%	90.46%	97.95%

Table 6.2.1.28 Week 18 Performance Metrics

These metrics were critical in identifying not just accuracy, but also the trade-offs between catching all anomalies and avoiding false alarms. Additionally, AUROC and AUPR scores were extracted from the training phase and included as baseline model quality indicators.

5. Visual Evaluation Insight

Week 18 visual inspection presented a full challenge with the biggest image set so far which had densely planted ginger crops lined up next to one another with minimal or no background distractions. This specific arrangement gave a cleaner environment for the detection of visual anomalies so clearer analysis could be performed of model behaviour under ideal conditions. Anomaly overlays were tested in full-resolution images, helping identify the strengths and weaknesses of each model.

- CFlow had unstable behavior. On well-behaved samples, it often produced streaky heatmaps with sparse false positives, reducing interpretability. Even when there was a clean background, the model struggled to maintain stability and tended to mark out non-anomalous regions. However, in dying and hue-down situations, while it could not produce robust anomaly masks, it rankably

identified the regions correctly—i.e., it could be utilized as a validation layer, but not as a primary detector.

- FastFlow delivered mean visual quality, particularly on normal samples where its output was acceptable despite numerous erroneous highlights. On dying and hue-down 30% samples, FastFlow was greatly improved, accurately highlighting regions of interest despite occasionally not extending to the anomaly boundary. However, a hue-down of 15% remained a soft spot, often not being highlighted. Overall, FastFlow's ability to localize anomalies made it visually comprehendible, but lacking in decisiveness about faint anomalies.

- PatchCore performed exceptionally well, especially on anomaly samples. It detected all dying symptoms confidently and generated consistent heatmaps with good localization. On great samples, the model flagged some residual pot structures as anomalies but successful filtering reduced most such false alarms. Its sensitivity was very useful in dying and hue-down testing. Despite some low-level misclassifications, PatchCore was one of the most dependable detectors for Week 18.

- Reverse Distillation was very sensitive, detecting both plant roots and subtle colour changes. Interestingly, it tended to flag triangular root growth patterns as anomalies, which are likely normal variations not well represented in the training data. On dead samples, even when detection scores were below the threshold, the anomaly regions were flagged—suggesting that with threshold tuning, performance could be significantly improved. Hue-down performance at 15% remained poor, but detection on structurally different features like roots was always robust.

- STFPM was still the most visually accurate model. On normal samples, it had no qualms about producing clean, blank masks. For dying leaves, it ranked anomaly regions with high confidence correctly, even for samples that technically failed detection thresholds. Hue-down 15% changes caused score reductions, but the anomaly regions were still distinctly highlighted, confirming

STFPM's localization capability. At 30%, detection became much clearer. Its ability to consistently favour correct areas, even under slight changes, makes STFPM the strongest and most consistent model visually.

In general, the Week 18 dataset (high size and low background noise) facilitated a purer test of anomaly detection performance. Highly context-dependent background models (e.g., CFlow) did not work well, whereas highly internally consistent models (PatchCore, STFPM) worked well. Hue-down transformations remained difficult for most but a handful of models at 15%, though higher variation levels improved interpretability. Visual intuition this week decidedly supports the use of STFPM and PatchCore in real-world applications, especially where accurate localization and low background interference are a top priority.

The Week 18 visual testing illustrates strong model robustness growth, especially against the high image count of the dataset and low background noise by highly dense plant stands. CFlow exhibited basic performance on shared samples but remained behind in anomaly cases with scant visual cues even when they contained blatant defects. FastFlow had decent consistency, accurately localizing bad sample errors, but also provided some bad classifications on good samples, showing ongoing sensitivity limitations. PatchCore had strong overall performance, particularly in dead leaf detection and rejecting irrelevant pot-based outliers, and is rated to be one of the better-performing models this week. Reverse Distillation performed well, especially where root abnormalities occurred in plants, although it tended to incorrectly flag normal root shapes as abnormal on occasion due to lack of exposure during training. STFPM again performed exceptionally well, correctly identifying gross and fine abnormalities, and even when detection failed, it still produced high anomaly scores for the right regions. While hue-down conversions, especially at 15%, remained a weak link for all models, the more explicit visual organization of Week 18 images further exposed each model's detection rationale. These findings reinforce the need for clean input data and continue to emphasize the need for diverse training samples and higher model sensitivity to tiny colour and structural changes.

6.2.2 Project Workflow Overview

To give a summary of the performance analysis, the table below is a summary of each model's average scores on all the test weeks and anomaly conditions. The measures that encompass

Accuracy, F1 Score, Specificity, and corresponding diagnostic measures are summarized. From the table, a glimpse of the effectiveness of each model can be seen as well as a comparison of overall model quality.

Model Performance Overview

Based on averaged performance metrics for all test setups, STFPM is overall the best model. It has very high sensitivity and specificity the best F1 Score (84.03%) and maximum detection power per false positive control. PatchCore is second-best with perfect precision and specificity but with marginally lower sensitivity and recall. Reverse Distillation ranks third with perfect performance in the majority of classes but with intermittent sensitivity sacrifices. Fourth is FastFlow, with decent overall accuracy but poor anomaly sensitivity and hue-based transformation. Last but not least, CFlow consistently underperformed, particularly in cases of dying and hue-down conditions, and hence is the least accurate among the models tested.

Appendix A shows the complete table.

model	cflow	fastflow	patchcore	reverse distillation	stfpm
Normal	74.90%	87.92%	100.00%	93.06%	100.00%
contrast down 0.7	65.97%	93.65%	100.00%	92.36%	93.75%
contrast down 0.9	78.33%	91.67%	100.00%	96.67%	100.00%
contrast up 1.1	71.88%	90.93%	100.00%	85.42%	100.00%
contrast up 1.3	71.67%	86.25%	100.00%	87.24%	100.00%
dying 1	47.53%	70.09%	83.01%	78.72%	74.11%
dying 2	22.44%	84.74%	100.00%	72.92%	95.64%
dying 3	33.19%	87.64%	100.00%	91.67%	100.00%
hue down 15	19.98%	24.30%	18.53%	29.13%	29.18%
hue down 30	25.00%	38.13%	42.50%	51.70%	79.39%
Accuracy	51.04%	73.49%	81.81%	77.55%	86.22%
Precision	55.62%	87.39%	100.00%	87.38%	98.08%
Recall (Sensitivity)	29.90%	56.31%	63.63%	64.58%	73.86%
Specificity	72.19%	90.67%	100.00%	90.79%	98.75%
F1 Score	37.65%	67.75%	76.94%	73.71%	84.03%
False Positive Rate	27.81%	9.33%	0.00%	9.21%	1.25%
False Negative Rate	70.10%	43.69%	36.37%	35.42%	26.14%
Balanced Accuracy	51.04%	73.49%	81.81%	77.68%	86.30%
Youden's Index (J)	2.09%	46.98%	63.63%	55.36%	72.61%
Negative Predictive Value	49.96%	67.66%	74.06%	72.36%	79.28%
AUROC	45.50%	78.86%	82.75%	81.06%	82.63%

AUPR	69.26%	79.95%	84.29%	81.83%	80.89%

Table 6.2.2.1 Average Result on Variable and Performance Metrics

- STFPM – Best Overall Performer

  Strengths:

  - Highest average accuracy (86.22%) and best F1 score (84.03%), showing a strong balance between precision and recall.
  - Excellent at localizing anomalies, particularly in dying and contrast variation scenarios.
  - High specificity (98.75%) and very low false positive rate (1.25%), meaning it rarely misclassifies healthy plants.
  - Best performer in hue-down 30%, a previously difficult category for all models.

  Weaknesses:

  - Slight sensitivity to minor hue shifts (15%), occasionally flagging healthy images.
  - May require tuning of sensitivity thresholds to prevent false alarms in borderline cases.

- PatchCore – Strong Precision, Slight Recall Limitations

  Strengths:

  - Perfect precision (100%) and specificity (100%), meaning it only flags anomalies when it is very sure.
  - Excellent in detecting dying conditions, particularly in “dying 2” and “dying 3”.
  - Ideal for applications that require minimal false positives.

  Weaknesses:

  - Moderate recall (63.63%), indicating it sometimes misses subtle anomalies.
  - Underperformed slightly in hue-down scenarios, where sensitivity was not high enough to cross anomaly thresholds.
  - May benefit from greater sensitivity in early-stage or subtle color changes.

- Reverse Distillation – Balanced and Promising

  Strengths:

- Consistently strong across many conditions, especially in contrast variations and dying 3.
- Good recall (64.58%) and F1 score (73.71%), indicating a balance between sensitivity and precision.
- Excellent at detecting structural abnormalities and discoloration, including root shape anomalies.

Weaknesses:

- Slight tendency to flag normal plant structures as anomalies, such as triangular roots, possibly due to limited diversity in training data.
- High variability in performance depending on visual input complexity.

- FastFlow – Decent Generalist, Lacking Sensitivity

  Strengths:

  - High precision (87.39%) and specificity (90.67%), indicating a good ability to avoid false positives.
  - Performs well in normal conditions and under mild contrast variations.

  Weaknesses:

  - Low recall (56.31%) and limited sensitivity to subtle anomalies like early discoloration or mild dying.
  - Poor performance in hue-down conditions, especially at 15%.
  - Often detects the anomaly regions visually but fails to surpass detection thresholds.

- CFlow – Least Reliable Overall

  Strengths:

  - Performs acceptably on normal samples (74.90% accuracy), often generating clean maps with minimal false positives.
  - Good at indicating normalcy rather than flagging anomalies, making it suitable as a baseline validator.

  Weaknesses:

- Lowest performance across nearly all metrics, including recall (29.90%), F1 score (37.65%), and AUROC (45.50%).
- Poor sensitivity to hue-down and dying samples, often failing to flag visible anomalies.
- May lack robustness against color-based transformations or require better threshold tuning.

This overview supports the requirement of model selection based on overall as well as task-dependent factors, especially for agriculture anomaly detection where sensitivity to marginal visual changes becomes a mandate.

6.2.3 Observation Paragraph

During the test weeks, the top-performing week overall was Week 12, during which most of the models were found to be highly consistent with visual anomalies with the highest possible detection confidence, particularly for the dying and contrast transformation scenarios. Throughout the course of the study, STFPM was uniformly superior to other models, producing the most consistent anomaly localization, the highest average accuracy, and a balanced sensitivity-specificity profile. PatchCore similarly exhibited exceptional precision, especially in false negative avoidance, and Reverse Distillation presented a balanced and flexible performance under varying conditions. CFlow, by contrast, uniformly underperformed, especially in nuanced transformations, demonstrating shortfalls in its anomaly thresholding. Among all tested features, hue-based transformations (particularly hue-down 15%) posed the strongest challenge to all models, more often than not blurring the boundary between anomalous and healthy appearances. The challenge highlights the necessity of increasing colour sensitivity as well as enhanced feature calibration to further enhance anomaly detection robustness under real-world plant imagery.

6.3 Project Challenges

During the construction and testing of the ginger plant anomaly detection system, several problems of a practical and technical nature cropped up that influenced the system's reliability and construction timeline.

- Inconsistent Weekly Image Data: One of the most salient problems was inconsistency in weekly image recordings. Variations in camera orientation, illumination, and stability of recordings led data to differ radically week after week. Such inconsistency reduced model resistance and made it challenging to compare results reasonably over test times.

- Poor Image Framing: The majority of the images were small or snapped from an in-close angle, cropping the ginger plant. Since the models depend on full plant context to detect anomalies, these weakly framed inputs decreased detection performance and undermined the anomaly localization validity.

- Constraints on Hardware: Low computing capacity restricted the training process, particularly longer model training sessions. The limitations affected the training batch sizes and the possibility of experimenting with larger or more complex models, causing the progress to be slower and real-time testing to be limited.

- Limited Image Quantity: The data set contained too few images, limiting the model's learning capacity and generalizing performance across different plant condition types. In some weeks, there were too few available samples to be assessed validly.

- Lack of Anomalous Data: One of the large issues was that there were no annotated anomaly images present in the training data. This limited the models from learning actual-world disease symptoms and required them to rely on subtle statistical variances, which may not have been substantial.

- Training Data Affected Test Output: As the training input images did not cover all ranges of plant orientations and environmental conditions, test outputs were highly sensitive to direction or position changes. This difference reduced performance consistency and emphasized diversified training sets.

- Time-Consuming Training Process: Training of the anomaly detection model, although on a small dataset, was time-consuming—particularly when tuning hyperparameters or

retraining after modification. This slowed down the process of evaluation and the number of experiments possible within the given timeframe

Despite all these challenges, the project managed to yield valuable insights into the performance and abilities of unsupervised anomaly detection models for monitoring plant health.

6.4 Objectives Evaluation

This section examines the extent to which the project met the initial goals defined at the beginning, considering both the technical implementation and the practical issues faced.

- Objectives

1. Objective 1: Detection of Anomalous Ginger Plant was partially achieved. The model was able to classify plants as normal or anomalous and could localize the anomalous regions with heat maps. However, despite models like STFPM and PatchCore performing well, the detection accuracy was image condition-dependent. False positives and localization mistakes were seen, especially in suboptimal test conditions.

2. Objective 2: A Practical Health Monitoring System was achieved to some extent. The system was tested with weekly plantation data gathered from a real ginger farm. While the detection models ran on real-world inputs, inconsistency in images, insufficient anomaly samples, and small dataset sizes impacted performance consistency. The system had promise but needs improvement to be consistent in general field conditions.

3. Objective 3: Unsupervised Anomaly Detection was fully achieved. All the models tested were trained unsupervised on healthy plant images. Five state-of-the-art unsupervised AD models (CFlow, FastFlow, PatchCore, Reverse Distillation, STFPM) were tested in the project and compared for anomaly detection performance without prior annotation of anomalies.

As discussed in Section 6.3, several factors prevented the full achievement of objectives. Poor quality and inconsistent image data prevented the generalizability of models across test environments. Lack of proper confirmation of anomaly samples restricted testing against true

performance, while time and hardware limitations prevented extensive tuning and experimenting. These factors affected the achievement of Objectives 1 and 2 most.

6.5 Concluding Remark

- Summary of Findings

  The project managed to effectively validate the applicability of using unsupervised anomaly detection models in ginger plant health monitoring. STFPM and PatchCore were among the five tested models that performed best and consistently with robust and high performance across varying test conditions, particularly excelling others in precision and locality of anomalies. The performance of the system was, however, affected by real-world issues such as irregularly occurring weak image quality, limited anomaly samples, and limited dataset size. These problems made it difficult to ensure consistent detection accuracy, especially in ambiguous cases like weak colouration or image distortions.

- Lessons Learned

  Several valuable lessons were uncovered in the process of the project. Data quality was as crucial as model choice—issues like bad framing, image resolution, and limited anomalies had a tremendous effect on model performance. It also became clear that even the best models require meticulous preprocessing and evaluation plans to be able to perform effectively in real-world agricultural environments. Additionally, coordinating the training and evaluation process within time and hardware constraints was an exercise in learning resource optimization and project scope management.

- System Success and Limitations

  Overall, the system succeeded in its primary technical objectives, showing that unsupervised AD models can detect anomalies in real ginger plantation data without labelling. Integration with a user interface like a Discord bot and successful deployment of Anomalib-based models were significant achievements. Despite these efforts, the system remains susceptible to limitations—primarily, reliance on good-quality input images, challenges in detecting minor anomalies, and inadequate model robustness in extreme environments. Unavailability of confirmed anomaly samples also limited capability to quantify true model accuracy under field conditions.

- Suggestions for Future Work

  Future development should focus on getting the dataset more uniform and highly resolved by employing additional images and obtaining validated instances of anomalies. Tuning the hyperparameters for the model, exploring the lightweight deployment of the model for real-time use, and improving image processing techniques can continue to improve the accuracy. Getting the system to support operation with multiple plant types, incorporating active learning for enhanced annotation of anomalies, and supporting user interface functionality (e.g., mobile or web-based dashboards) would both expand the useful application and extensibility of the system.)

Chapter 7
Conclusion and Recommendation

This chapter concludes the research and development process of building an unsupervised anomaly detection system for ginger plant health monitoring. It recapitulates the primary discoveries, discusses the system's overall performance, and provides closure to the primary objectives of the project. The reader can expect an overview of technical breakthroughs, system performance, limitations faced, and the overall implications of the discoveries on agricultural technology and avenues for further research.

7.1 Conclusion

This project was able to explore the use of an unsupervised anomaly detection system for monitoring the health of ginger plants using real farming data. Through the use of state-of-the-art models provided by the Anomalib framework, the system was able to evaluate the plant conditions with promising outcomes on different environmental conditions and weekly image batches. Among the models tried out—CFlow, FastFlow, PatchCore, Reverse Distillation, and STFPM—STFPM and PatchCore were the most stable performers, with high specificity, decent precision, and stable anomaly localization even for visually challenging cases.

The integration of anomaly detection and an easy-to-use Discord bot interface made the system usable and feasible for potential real-world adoption. However, limitations such as variable image quality, limited anomaly samples, and time-demanding training processes impacted the robustness and generalizability of the solution. Despite these limitations, the system showed promise as a non-invasive, automated ginger plant health monitoring system under field conditions.

In conclusion, the project achieved its main objectives: implementing an unsupervised detection model, creating a prototype real-life health monitoring system, and evaluating its effectiveness through diverse test scenarios. The results stress the importance of quality datasets and standard preprocessing in achieving accurate anomaly detection in agricultural environments..

7.2 Recommendation

Based on the observations and challenges encountered throughout this project, several recommendations are provided to ensure better development and deployment in the future:

1. Improve Image Data Collection

   Ensure image datasets are always captured with good framing, lighting, and resolution. Avoid very close or zoomed-in shots that crop out parts of the plant, and try to have consistency in the conditions for each weekly collection.

2. Expand the Dataset

   Increase the size and diversity of the training dataset, particularly by including confirmed anomaly cases (e.g., diseased, wilted, pest-affected plants). This will help improve model reliability and allow for more meaningful evaluation metrics.

3. Enhance Preprocessing Pipelines

   Implement preprocessing techniques such as background removal, colour normalization, and image stabilization to increase model input quality. These steps can provide a significant boost to detection accuracy, particularly for subtle anomalies.

4. Optimize Training Efficiency

   Reduce the training time by exploring more computationally efficient variants of the models or transfer learning approaches. Explore GPU acceleration and batch training approaches to make the system more scalable.

5. Extend to Real-Time Monitoring

   Explore lightweight model versions for real-time applications so that live feedback through the Discord bot or other mobile/web interfaces is possible. This would significantly enhance the ease of use of the system for farmers.

6. Consider Semi-Supervised Learning

   Introduce semi-supervised methods that can include limited labeled data to direct learning, with the potential to enhance performance in situations where fully unsupervised techniques fail.

7. Collaborate with Agricultural Experts

   Closely interact with farmers or agronomists in order to confirm the anomalies identified by the system and make the outputs actionable and meaningful within an actual farming setup.

These recommendations are aimed at developing the prototype into a robust, scalable, and efficient agricultural monitoring system. With further development and incorporation of additional data, the system can significantly assist farmers in the early identification of plant health issues, improving yield and sustainability in ginger farming.

REFERENCES

[1] The Editors of Encyclopedia Britannica, “Ginger | plant,” Encyclopædia Britannica. Jan. 11, 2019. Available: https://www.britannica.com/plant/ginger

[2] “How To Grow Ginger? Growing Ginger Root Is Not That Hard...,” Tropicalpermaculture.com, 2020. https://www.tropicalpermaculture.com/growing-ginger.html

[3] “Smart Ginger Cultivation Techniques: Revolutionizing Rhizome Production in Indonesia - FnB Tech,” FnB Agritech, Jul. 16, 2024. https://agritech.fnb.tech/smart-ginger-cultivation-techniques-production/ (accessed Sep. 02, 2024).

[4] Aceng Sambas, Mujiarto Mujiarto, Gugun Gundara, Gunawan Refiadi, Neneng Sri Mulyati, and Ibrahim Mohammed Sulaiman, “Development of Smart Farming Technology on Ginger Plants in Padamulya Ciamis Village, West Java, Indonesia,” International Journal of Research in Community Service, vol. 4, no. 3, pp. 93–99, Jul. 2023, doi: https://doi.org/10.46336/ijrcs.v4i3.483.

[5] Britannica.com, 2024. https://cdn.britannica.com/19/231119-050-35483892/Indian-ginger-Zingiber-officinale.jpg (accessed Sep. 07, 2024).

[6] Hawaii.edu, 2024. https://cms.ctahr.hawaii.edu/portals/43/Ginger%20Field.jpg (accessed Sep. 07, 2024).

[7] Peat-cloud.com, 2024. https://content.peat-cloud.com/w400/leaf-spot-of-ginger-ginger-1582821332.jpg (accessed Sep. 07, 2024).

[8] Zenadrone.com, 2024. https://www.zenadrone.com/wp-content/uploads/2022/10/smart-farming-and-plantation.jpg (accessed Sep. 07, 2024).

[9] S. C M and C. Raju, "Revolutionizing Crop Management: An Emphasis on Ginger Leaf Disease Detection Techniques Using Machine Learning and IoT," 2023 International Conference on Data Science and Network Security (ICDSNS), Tiptur, India, 2023, pp. 1-5, doi: 10.1109/ICDSNS58469.2023.10245472.

[10] J. Boulent, S. Foucher, J. Théau, and P.-L. St-Charles, “Convolutional Neural Networks for the Automatic Identification of Plant Diseases,” Frontiers in Plant Science, vol. 10, Jul. 2019, doi: https://doi.org/10.3389/fpls.2019.00941.

[11] G. G. and A. P. J., “Identification of plant leaf diseases using a nine-layer deep convolutional neural network,” Computers & Electrical Engineering, vol. 76, pp. 323–338, Jun. 2019, doi: https://doi.org/10.1016/j.compeleceng.2019.04.011.

[12] D. Li et al., “A Recognition Method for Rice Plant Diseases and Pests Video Detection Based on Deep Convolutional Neural Network,” Sensors, vol. 20, no. 3, p. 578, Jan. 2020, doi: https://doi.org/10.3390/s20030578.

[13] “What is Anomaly Detection? Definition & FAQs,” Avi Networks. https://avinetworks.com/glossary/anomaly-detection/#:~:text=Anomaly%20detection%20is%20the%20identification

[14] Marcelinus A.S. Adhiwibawa, Waego Hadi Nugroho, and None Solimun, “Detection of Anomalies in Citrus Leaves Using Digital Image Processing and T2 Hotelling Multivariate Control Chart,” Mar. 2019, doi: https://doi.org/10.1109/icaiit.2019.8834453.

[15] C. Catalano, L. Paiano, F. Calabrese, M. Cataldo, L. Mancarella, and F. Tommasi, “Anomaly detection in smart agriculture systems,” Computers in Industry, vol. 143, p. 103750, Dec. 2022, doi: https://doi.org/10.1016/j.compind.2022.103750.

[16] A.P.Nirmala, Ansar Isak Sheikh, Dr. R. Kesavamoorthy, Dr. Raja M, Anantha Rao Gottimukkala, and Dr.R.Thiagarajan, “An Approach for Detecting Complications in Agriculture Using Deep Learning and Anomaly-Based Diagnosis,” Mathematical Statistician and Engineering Applications, vol. Vol 70 No. 2 (2021), no. 2094-0343, pp. 880–889, Dec. 2021, doi: https://doi.org/10.17762/msea.v70i2.2086.

[17] D. C. Ilie-Ablachim and B. Dumitrescu, “Angle-Based Dictionary Learning for Outlier Detection,” pp. 01–06, Dec. 2023, doi: https://doi.org/10.1109/scc59637.2023.10527594.

[18] H. Du, S. Zhao, D. Zhang, and J. Wu, “Novel clustering-based approach for Local Outlier Detection,” IEEE Xplore, Apr. 01, 2016. https://ieeexplore.ieee.org/document/7562187 (accessed May 30, 2023).

[19] Y. Wang, K. Li, and S. Gan, “A Kernel Connectivity-based Outlier Factor Algorithm for Rare Data Detection in a Baking Process,” IFAC-PapersOnLine, vol. 51, no. 18, pp. 297–302, 2018, doi: https://doi.org/10.1016/j.ifacol.2018.09.316.

[20] I. Aguilera-Martos et al., “Multi-step histogram based outlier scores for unsupervised anomaly detection: ArcelorMittal engineering dataset case of study,” Neurocomputing, vol. 544, pp. 126228–126228, Aug. 2023, doi: https://doi.org/10.1016/j.neucom.2023.126228.

[21] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation Forest,” 2008 Eighth IEEE International Conference on Data Mining, Dec. 2008, doi: https://doi.org/10.1109/icdm.2008.17.

[22] Y. Chen, Q. Zhao, and L. Lu, “Combining the outputs of various k-nearest neighbor anomaly detectors to form a robust ensemble model for high-dimensional geochemical anomaly detection,” Journal of Geochemical Exploration, vol. 231, p. 106875, Dec. 2021, doi: https://doi.org/10.1016/j.gexplo.2021.106875.

[23] A. Wijayanto, A. Sugiharto, and R. Santoso, “Detection Model for Potential Flooding Areas Using K-Means and Local Outlier Factor (LOF),” 2024 4th International Conference of Science and Information Technology in Smart Administration (ICSINTESA), pp. 445–450, Jul. 2024, doi: https://doi.org/10.1109/icsintesa62455.2024.10747854.

[24] J. ALMutawa, “Identification of errors-in-variables model with observation outliers based on Minimum-Covariance-Determinant,” Proceedings of the ... American Control

Conference/Proceedings of the American Control Conference, Jul. 2007, doi: https://doi.org/10.1109/acc.2007.4282931.

[25] Bas van Stein, Matthijs van Leeuwen, and T. Bäck, “Local subspace-based outlier detection using global neighbourhoods,” arXiv (Cornell University), Dec. 2016, doi: https://doi.org/10.1109/bigdata.2016.7840717.

[26] K. Chatterjee, K. Mahapatra, and N. R. Chaudhuri, “Robust Recovery of PMU Signals With Outlier Characterization and Stochastic Subspace Selection,” IEEE Transactions on Smart Grid, vol. 11, no. 4, pp. 3346–3358, Jul. 2020, doi: https://doi.org/10.1109/tsg.2019.2961561.

[27] T. Reiss and Y. Hoshen, “Attribute-based Representations for Accurate and Interpretable Video Anomaly Detection,” arXiv.org, Dec. 01, 2022. https://arxiv.org/abs/2212.00789 (accessed Nov. 05, 2023).

[28] S. Lee, S. Lee, and Byung Cheol Song, “CFA: Coupled-hypersphere-based Feature Adaptation for Target-Oriented Anomaly Localization,” arXiv (Cornell University), Jun. 2022, doi: https://doi.org/10.48550/arxiv.2206.04325.

[29] D. Gudovskiy, Shun Ishizaka, and Kazuki Kozuka, “CFLOW-AD: Real-Time Unsupervised Anomaly Detection with Localization via Conditional Normalizing Flows,” arXiv (Cornell University), Jul. 2021, doi: https://doi.org/10.48550/arxiv.2107.12571.

[30] M. Rudolph, T. Wehrbein, B. Rosenhahn, and B. Wandt, “Fully Convolutional Cross-Scale-Flows for Image-based Defect Detection,” arXiv (Cornell University), Jan. 2021, doi: https://doi.org/10.48550/arxiv.2110.02855.

[31] Vitjan Zavrtanik, M. Kristan, and Danijel Skočaj, “DRAEM -- A discriminatively trained reconstruction embedding for surface anomaly detection,” Aug. 2021, doi: https://doi.org/10.48550/arxiv.2108.07610.

[32] “DFKDE - Anomalib v0.3.7,” Readthedocs.io, 2023. https://anomalib.readthedocs.io/en/v0.3.7/reference_guide/algorithms/dfkde.html (accessed Dec. 03, 2024).

[33] N. A. Ahuja, I. Ndiour, T. Kalyanpur, and O. Tickoo, “Probabilistic Modeling of Deep Features for Out-of-Distribution and Adversarial Detection,” arXiv.org, Sep. 25, 2019. https://arxiv.org/abs/1909.11786 (accessed Jul. 18, 2023).

[34] V. Zavrtanik, M. Kristan, and D. Skočaj, “DSR -- A dual subspace re-projection network for surface anomaly detection,” arXiv (Cornell University), Jan. 2022, doi: https://doi.org/10.48550/arxiv.2208.01521.

[35] K. Batzner, L. Heckler, and R. König, “EfficientAD: Accurate Visual Anomaly Detection at Millisecond-Level Latencies,” arXiv.org, Mar. 25, 2023. https://arxiv.org/abs/2303.14535v1 (accessed May 26, 2023).

[36] J. Yu et al., “FastFlow: Unsupervised Anomaly Detection and Localization via 2D Normalizing Flows,” arXiv (Cornell University), Nov. 2021, doi: https://doi.org/10.48550/arxiv.2111.07677.

[37] I. Ndiour, N. Ahuja, U. Genc, and O. Tickoo, “FRE: A Fast Method For Anomaly Detection And Segmentation,” arXiv (Cornell University), Jan. 2022, doi: https://doi.org/10.48550/arxiv.2211.12650.

[38] S. Akcay, Amir Atapour-Abarghouei, and T. P. Breckon, “GANomaly: Semi-Supervised Anomaly Detection via Adversarial Training,” May 2018, doi: https://doi.org/10.48550/arxiv.1805.06725.

[39] T. Defard, Aleksandr Setkov, A. Loesch, and Romaric Audigier, “PaDiM: a Patch Distribution Modeling Framework for Anomaly Detection and Localization,” arXiv (Cornell University), Nov. 2020, doi: https://doi.org/10.48550/arxiv.2011.08785.

[40] K. Roth, Latha Pemula, J. Zepeda, Bernhard Schölkopf, T. Brox, and P. V. Gehler, “Towards Total Recall in Industrial Anomaly Detection,” arXiv (Cornell University), Jun. 2021, doi: https://doi.org/10.48550/arxiv.2106.08265.

[41] H. Deng and X. Li, “Anomaly Detection via Reverse Distillation from One-Class Embedding,” arXiv.org, 2022, doi: https://doi.org/10.48550/arXiv.2201.10703.

[42] P. Adey, O. Hamilton, Magnus Bordewich, and T. Breckon, “Region Based Anomaly Detection with Real-Time Training and Analysis,” 2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 495–499, Dec. 2019, doi: https://doi.org/10.1109/icmla.2019.00092.

[43] G. Wang, S. Han, E. Ding, and D. Huang, “Student-Teacher Feature Pyramid Matching for Anomaly Detection,” arXiv (Cornell University), Mar. 2021, doi: https://doi.org/10.48550/arxiv.2103.04257.

[44] M. Tailanian, Á. Pardo, and P. Musé, “U-Flow: A U-shaped Normalizing Flow for Anomaly Detection with Unsupervised Threshold,” arXiv (Cornell University), Jan. 2022, doi: https://doi.org/10.48550/arxiv.2211.12353.

[45] J. Jeong, Y. Zou, T. Kim, D. Zhang, A. Ravichandran, and O. Dabeer, “WinCLIP: Zero-/Few-Shot Anomaly Classification and Segmentation,” arXiv.org, Mar. 26, 2023. https://arxiv.org/abs/2303.14814 (accessed Jun. 12, 2024).

[46] G. Xu, P. Jin, L. Hao, Y. Song, L. Sun, and L. Yuan, “LLaVA-o1: Let Vision Language Models Reason Step-by-Step,” arXiv (Cornell University), Nov. 2024, doi: https://doi.org/10.48550/arxiv.2411.10440.

[47] M. Abdin et al., “Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone,” arXiv.org, Apr. 23, 2024. https://arxiv.org/abs/2404.14219

[48] A. Dubey et al., “The Llama 3 Herd of Models,” arXiv (Cornell University), Jul. 2024, doi: https://doi.org/10.48550/arxiv.2407.21783.

[49] A. Ghosh, A. Acharya, S. Saha, V. Jain, and A. CHadha, “Exploring the Frontier of Vision-Language Models: A Survey of Current Methodologies and Future Directions,” arXiv (Cornell University), Feb. 2024, doi: https://doi.org/10.48550/arxiv.2404.07214.

[50] S. Hu et al., “MiniCPM: Unveiling the Potential of Small Language Models with Scalable Training Strategies,” arXiv (Cornell University), Apr. 2024, doi: https://doi.org/10.48550/arxiv.2404.06395.

[51] JiaWeiBu, “GingerPlantAnomaly/classes/README.md at main · JiaWeiBu/GingerPlantAnomaly,” GitHub, 2025. https://github.com/JiaWeiBu/GingerPlantAnomaly/blob/main/classes/README.md (accessed May 03, 2025).

[52] JiaWeiBu, “GitHub - JiaWeiBu/GingerPlantAnomaly: A unsupervised ginger plant anomaly detection model” GitHub, 2025. https://github.com/JiaWeiBu/GingerPlantAnomaly (accessed May 03, 2025).

[53] JiaWeiBu, “GingerPlantAnomaly/README_setup.md at main · JiaWeiBu/GingerPlantAnomaly,” GitHub, 2025. https://github.com/JiaWeiBu/GingerPlantAnomaly/blob/main/README_setup.md (accessed May 03, 2025).

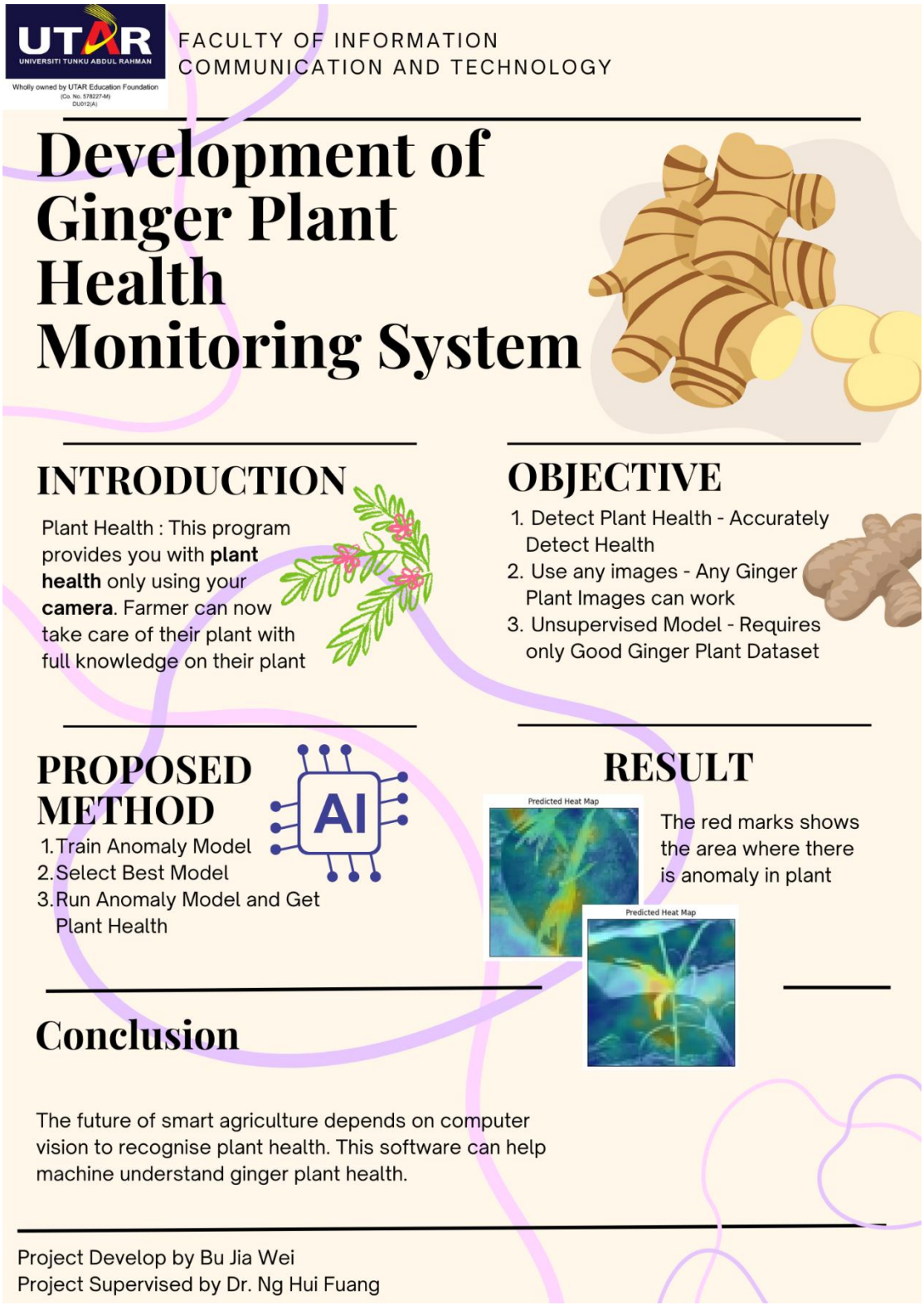
APPENDIX A Training Result

variable	week 3					week 8				
model	cflow	fastflow	patchcore	reverse di:	stfpm	cflow	fastflow	patchcore	reverse di:	stfpm
Normal	83.33%	100.00%	100.00%	83.33%	100.00%	100.00%	91.67%	100.00%	100.00%	100.00%
contrast down 0.7	75.00%	100.00%	100.00%	100.00%	75.00%	78.57%	100.00%	100.00%	100.00%	100.00%
contrast down 0.9	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
contrast up 1.1	75.00%	100.00%	100.00%	75.00%	100.00%	81.82%	90.91%	100.00%	100.00%	100.00%
contrast up 1.3	66.67%	100.00%	100.00%	66.67%	100.00%	100.00%	80.00%	100.00%	92.31%	100.00%
dying 1	100.00%	100.00%	75.00%	75.00%	50.00%	27.27%	72.73%	81.82%	87.50%	75.00%
dying 2	0.00%	100.00%	100.00%	0.00%	100.00%	30.00%	100.00%	100.00%	91.67%	91.67%
dying 3	33.33%	66.67%	100.00%	66.67%	100.00%	11.11%	88.89%	100.00%	100.00%	100.00%
hue down 15	16.67%	0.00%	0.00%	0.333333	0.00%	30.77%	38.46%	15.38%	18.18%	0.454545
hue down 30	20.00%	20.00%	20.00%	33.33%	0.833333	20.00%	0.00%	30.00%	38.46%	0.692308
TN	16	20	20	17	19	48	50	53	52	54
FP	4	0	0	3	1	5	3	0	1	0
FN	13	11	11	12	9	40	22	20	19	13
TP	7	9	9	9	12	13	31	33	34	40
Accuracy	57.50%	72.50%	72.50%	63.41%	75.61%	57.55%	76.42%	81.13%	81.13%	87.85%
Precision	63.64%	100.00%	100.00%	75.00%	92.31%	72.22%	91.18%	100.00%	97.14%	100.00%
Recall (Sensitivity)	35.00%	45.00%	45.00%	42.86%	57.14%	24.53%	58.49%	62.26%	64.15%	75.47%
Specificity	80.00%	100.00%	100.00%	85.00%	95.00%	90.57%	94.34%	100.00%	98.11%	100.00%
F1 Score	45.16%	62.07%	62.07%	54.55%	70.59%	36.62%	71.26%	76.74%	77.27%	86.02%
False Positive Rate	20.00%	0.00%	0.00%	15.00%	5.00%	9.43%	5.66%	0.00%	1.89%	0.00%
False Negative Rate	65.00%	55.00%	55.00%	57.14%	42.86%	75.47%	41.51%	37.74%	35.85%	24.53%
Balanced Accuracy	57.50%	72.50%	72.50%	63.93%	76.07%	57.55%	76.42%	81.13%	81.13%	87.74%
Youden's Index (J)	15.00%	45.00%	45.00%	27.86%	52.14%	15.09%	52.83%	62.26%	62.26%	75.47%
Negative Predictive Value	55.17%	64.52%	64.52%	58.62%	67.86%	54.55%	69.44%	72.60%	73.24%	80.60%
AUROC	81.00%	68.50%	79.25%	76.75%	75.75%	84.75%	75.99%	76.68%	84.12%	85.62%	80.69%
AUPR	82.69%	72.28%	83.09%	76.45%	81.79%	84.75%	75.99%	89.04%	88.83%	77.17%

week 12						week 18					
cflow	fastflow	patchcore	reverse di	stfpm		cflow	fastflow	patchcore	reverse di	stfpm	
60.00%	60.00%	100.00%	88.89%	100.00%		56.25%	100.00%	100.00%	100.00%	100.00%	
71.43%	85.71%	100.00%	75.00%	100.00%		38.89%	88.89%	100.00%	94.44%	100.00%	
80.00%	80.00%	100.00%	100.00%	100.00%		33.33%	86.67%	100.00%	86.67%	100.00%	
83.33%	83.33%	100.00%	66.67%	100.00%		47.37%	89.47%	100.00%	100.00%	100.00%	
80.00%	80.00%	100.00%	100.00%	100.00%		40.00%	85.00%	100.00%	90.00%	100.00%	
20.00%	60.00%	80.00%	71.43%	85.71%		42.86%	47.62%	95.24%	80.95%	85.71%	
14.29%	57.14%	100.00%	100.00%	100.00%		45.45%	81.82%	100.00%	100.00%	90.91%	
33.33%	100.00%	100.00%	100.00%	100.00%		55.00%	95.00%	100.00%	100.00%	100.00%	
20.00%	0.4	0.4	0.4	0.4		12.50%	18.75%	18.75%	25.00%	0.3125	
25.00%	37.50%	25.00%	0.75	0.75		35.00%	0.95	0.95	60.00%	90.00%	
21	22	28	24	28		38	79	88	83	88	
7	6	0	4	0		50	9	0	5	0	
22	13	10	6	5		54	28	15	24	17	
6	15	18	22	23		34	60	73	64	71	
48.21%	66.07%	82.14%	82.14%	91.07%		40.91%	78.98%	91.48%	83.52%	90.34%	
46.15%	71.43%	100.00%	84.62%	100.00%		40.48%	86.96%	100.00%	92.75%	100.00%	
21.43%	53.57%	64.29%	78.57%	82.14%		38.64%	68.18%	82.95%	72.73%	80.68%	
75.00%	78.57%	100.00%	85.71%	100.00%		43.18%	89.77%	100.00%	94.32%	100.00%	
29.27%	61.22%	78.26%	81.48%	90.20%		39.53%	76.43%	90.68%	81.53%	89.31%	
25.00%	21.43%	0.00%	14.29%	0.00%		56.82%	10.23%	0.00%	5.68%	0.00%	
78.57%	46.43%	35.71%	21.43%	17.86%		61.36%	31.82%	17.05%	27.27%	19.32%	
48.21%	66.07%	82.14%	82.14%	91.07%		40.91%	78.98%	91.48%	83.52%	90.34%	
-3.57%	32.14%	64.29%	64.29%	82.14%		-18.18%	57.95%	82.95%	67.05%	80.68%	
48.84%	62.86%	73.68%	80.00%	84.85%		41.30%	73.83%	85.44%	77.57%	83.81%	
7.14%	78.83%	75.77%	73.09%	76.59%		17.88%	91.43%	91.86%	88.78%	97.48%	
75.00%	78.49%	70.91%	71.60%	66.63%		34.60%	93.03%	94.11%	90.46%	97.95%	

Average					
cflow	fastflow	patchcore	reverse dist	stfpm	
74.90%	87.92%	100.00%	93.06%	100.00%	
65.97%	93.65%	100.00%	92.36%	93.75%	
78.33%	91.67%	100.00%	96.67%	100.00%	
71.88%	90.93%	100.00%	85.42%	100.00%	
71.67%	86.25%	100.00%	87.24%	100.00%	
47.53%	70.09%	83.01%	78.72%	74.11%	
22.44%	84.74%	100.00%	72.92%	95.64%	
33.19%	87.64%	100.00%	91.67%	100.00%	
19.98%	24.30%	18.53%	29.13%	29.18%	
25.00%	38.13%	42.50%	51.70%	79.39%	
31	43	47	44	47	
17	5	0	3	0	
32	19	14	15	11	
15	29	33	32	37	
51.04%	73.49%	81.81%	77.55%	86.22%	
55.62%	87.39%	100.00%	87.38%	98.08%	
29.90%	56.31%	63.63%	64.58%	73.86%	
72.19%	90.67%	100.00%	90.79%	98.75%	
37.65%	67.75%	76.94%	73.71%	84.03%	
27.81%	9.33%	0.00%	9.21%	1.25%	
70.10%	43.69%	36.37%	35.42%	26.14%	
51.04%	73.49%	81.81%	77.68%	86.30%	
2.09%	46.98%	63.63%	55.36%	72.61%	
49.96%	67.66%	74.06%	72.36%	79.28%	
45.50%	78.86%	82.75%	81.06%	82.63%	
69.26%	79.95%	84.29%	81.83%	80.89%	

POSTER


UTAR
UNIVERSITI TUNKU ABDUL RAHMAN
Wholly owned by UTAR Education Foundation
FACULTY OF INFORMATION
COMMUNICATION AND TECHNOLOGY
Development of Ginger Plant Health Monitoring System
INTRODUCTION
Plant Health : This program provides you with plant health only using your camera. Farmer can now take care of their plant with full knowledge on their plant
OBJECTIVE
1. Detect Plant Health - Accurately Detect Health
2. Use any images - Any Ginger Plant Images can work
3. Unsupervised Model - Requires only Good Ginger Plant Dataset
PROPOSED METHOD
AI
1. Train Anomaly Model
2. Select Best Model
3. Run Anomaly Model and Get Plant Health
RESULT
Predicted Heat Map
Predicted Heat Map
The red marks shows the area where there is anomaly in plant
Conclusion
The future of smart agriculture depends on computer vision to recognise plant health. This software can help machine understand ginger plant health.
Project Develop by Bu Jia Wei
Project Supervised by Dr. Ng Hui Fuang