

SMART SURVEILLANCE SYSTEM FOR COCONUT DISEASES AND INFESTATIONS

Samitha Peruma Vidhanaarachchi

IT18078510

B.Sc. (Hons) Degree in Information Technology Specialized in Software
Engineering

Department of Computer Science and Software Engineering

Sri Lanka Institute of Information Technology
Sri Lanka

October 2021

SMART SURVEILLANCE SYSTEM FOR COCONUT DISEASES AND INFESTATIONS

Samitha Peruma Vidhanaarachchi

IT18078510

Dissertation submitted in partial fulfillment of the requirements for the Special Honours
Degree of Bachelor of Science in Information Technology Specializing in Software
Engineering


Department of Computer Science and Software Engineering

Sri Lanka Institute of Information Technology
Sri Lanka

October 2021

DECLARATION

I declare that this is my own work and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Name	Student ID	Signature
S.P Vidhanaarachchi	IT18078510	

Signature of the Supervisor
(Dr. Janaka Wijekoon)

Date

.....

.....

ABSTRACT

Coconut products are gaining popularity in the world market majorly among health-conscious consumers. A range of new value added products that were trade globally in the recent years show an increasing export volume. Therefore, coconut growing countries are developing strategies to increase the production and productivity to meet the increasing demand. One of the major challenges for coconut production is the yield loss due to pest and disease outbreaks. During last few decades many control methods were identified for management of various pest and diseases of coconut. Application of these control methods at the farm level was hindered due to poor communication between research personals and growers. Smart ICT intervention could play a major role in bridging this gap to develop efficient pest and disease management strategies in coconut. The coconut caterpillar is one of the major pests of coconut for which control methods have been developed. Due to the difficulty of identification of symptoms, out breaks occur leading to economic yield loss. The aim of the research is to develop a mobile application identification process to classify coconut caterpillar infestation by deep convolutional neural network techniques. Furthermore, the progression level is also calculated to measure the infestation severity. Fast R-CNN is used for the identification while Mask R-CNN is used to calculate the progression area. Number of caterpillars are calculated with both object detection (YOLO) and image processing techniques using OpenCV and TensorFlow libraries to help researchers decide on the correct control measures.

Keywords – Coconut caterpillar infestation, Mask R-CNN, YOLOv5, Image Processing

ACKNOWLEDGEMENT

First and foremost, I would like to express my sincere gratitude to my supervisor Dr. Janaka Wijekoon for the constant guidance and support which helped me at all times for the successful completion of my undergraduate research. Besides my supervisor, my sincere thanks also goes to Miss Dilani Lunugallage, the co-supervisor of this research project for being willing to help whenever it was needed. This research study being a mixture of technology and agriculture required the guidance and assistance of not only technology experts but also Agriculture professionals. The immense support extended by Dr. Mrs. Nyanie Aratchige throughout the project to bridge the knowledge gap in those areas is highly appreciated. My sincere gratitude also extends to Dr. Sarath Idirisinghe, Mr. Roshan De Silva, Mr. Prabhath Manoj, Mr. Anura Pathirana of Coconut Research Institute for their kind assistance extended by assisting identification of pest and disease symptoms of coconut. Last but not least, I express my sense of gratitude to my team mates, family, friends, to one and all, who directly or indirectly have extended their support throughout this project.

TABLE OF CONTENTS

DECLARATION	i
ABSTRACT	ii
ACKNOWLEDGEMENT	iii
LIST OF FIGURES	vi
LIST OF TABLES	vii
LIST OF ABBREVIATIONS	ix
1. INTRODUCTION	1
1.1 General Introduction	1
1.2 Background literature	2
1.2.1 An overview on coconut production in Sri Lanka	2
1.2.2 Pest and diseases of coconut	3
1.2.3 Deep learning models for pest and disease diagnosis	7
1.2 Research Gap	10
2. RESEARCH PROBLEM	13
3. RESEARCH OBJECTIVES	15
3.1 Main Objectives	15
3.2 Specific Objectives	15
4. METHODOLOGY	17
4.1 Materials and methods	17
4.1.1 Problem statement	18
4.1.2 Component System Architecture (Solution Design)	19
4.1.3 Data Acquisition and processing	20
4.1.4 Infestation identification and classification	22
4.1.5 Identification of the Degree of Diseased Condition in Leaves	28
4.1.6 Identification and counting caterpillars	31
4.1.7 Design Diagrams	41
4.2 Commercialization aspects of the product	43
5. Testing & Implementation	44
5.1 Implementation	44
5.1.1 Data Preprocessing	44
5.1.2 Data Augmentation	45
5.1.3 Deep learning model implementation	48
5.2 Testing	57

5.2.1 Test Plan and Test Strategy.....	57
5.2.2 Test Case Design	58
6. RESULTS AND DISCUSSIONS.....	64
6.1 Results.....	64
6.1.1 Infestation identification and classification	64
6.1.2 Identification of the Degree of Diseased Condition in Leaves	65
6.1.3 Identification and counting caterpillars (YOLOv5).....	67
6.1.4 Identification and counting caterpillars (Image Processing).....	69
6.1.4 Results of evaluating deep learning models.....	70
6.1.5 Product deployment	73
6.2 Research Findings.....	74
6.3 Discussion.....	75
7. CONCLUSIONS	77
8. REFERENCES	78
9. APPENDICES	84

LIST OF FIGURES

Figure 1. 1 : Different progression state	5
Figure 1. 2: Damage caused by Coconut Caterpillar [25].....	6
Figure 1. 3: Sample sheet of how data is collected for coconut caterpillar infestation	7
Figure 1. 4: Survey report on reasons for browning in coconut leaves	9
 Figure 2. 1: Survey of awareness on coconut caterpillar infestation	13
Figure 2. 2: Summary of willingness to learn about the severity	14
 Figure 4. 1: Overview system diagram	17
Figure 4. 2: Overview of component diagram	19
Figure 4. 3: Collected sample images used to train the model	21
Figure 4. 4: Data annotation using VGG annotator	23
Figure 4. 5: Convolutional operation [33].	24
Figure 4. 6: Pooling operation [33].....	25
Figure 4. 7: An example of computing IoUs for various bounding boxes [35].....	26
Figure 4. 8: Region of Interest align (RoIAlign) method [36].....	27
Figure 4. 9: Mask R-CNN framework with damage degree computation	28
Figure 4. 10: Image segmentation techniques[37]	29
Figure 4. 11: K-means clustering process [39]	31
Figure 4. 12: Overview process of scanning the white paper using image processing....	32
Figure 4. 13: Applying Gaussian blur to a noisy image (rice).....	34
Figure 4. 14: Applying Canny edge detection	34
Figure 4. 15: Overview process of calculating caterpillars using image processing	35
Figure 4. 16: Applying thresholding and eroding	36
Figure 4. 17: Overview process of calculating caterpillars using image processing	37
Figure 4. 18: Data annotation using makesense.ai.....	38
Figure 4. 19: YOLOv5 network architecture [51].	40
Figure 4. 20: Sequence diagram of CCI component.....	42
Figure 4. 21: Sequence diagram of CCI component.....	42
 Figure 5. 1: Steps of image preprocessing	44
Figure 5. 2: Code snippet for data preprocessing (resizing)	45
Figure 5. 3: Steps of data augmentation.....	46
Figure 5. 4: Code snippet for data augmentation (python)	47
Figure 5. 5: Code snippet for data augmentation (pseudocode)	47

Figure 5. 6: Augmentation techniques applied using Roboflow	48
Figure 5. 7: Steps of CCI identification and classification	48
Figure 5. 8: Cloning Github repository	49
Figure 5. 9: Code snippet for installing libraries needed for training	49
Figure 5. 10: Editing training configurations in “custom.py” file	50
Figure 5. 11: Code snippet to start the training process	50
Figure 5. 12: Created weight files	51
Figure 5. 13: Code snippet for loading the model	51
Figure 5. 14: Overview of progression level calculation	52
Figure 5. 15: Overview of caterpillar calculation using YOLOv5	53
Figure 5. 16: data.yaml file	54
Figure 5. 17: Code snippet used to create panoramic images	54
Figure 5. 18: Overview of caterpillar calculation image processing	55
Figure 5. 19: Code snippet to find the biggest contour and warp the image	56
Figure 5. 20: Code snippet used to find the connected components and calculate caterpillars	56
Figure 6. 1: Mask R-CNN instance segmentation and classification of leaflets	64
Figure 6. 2: Results of crop segmentation	65
Figure 6. 3: Results of colour segmentation	66
Figure 6. 4: Results of progression level calculation	66
Figure 6. 5: Summarized result of CCI classification and progression level calculation	67
Figure 6. 6: Results of caterpillar calculation using YOLOv5	68
Figure 6. 7: Results of creating panoramic image	68
Figure 6. 8: Results of creating stacked image	69
Figure 6. 9: Results of calculating caterpillars using image processing	70
Figure 6. 10: Precision graphs for Mask R-CNN and YOLOv5 models	71
Figure 6. 11: Recall graphs for Mask R-CNN and YOLOv5 models	71
Figure 6. 12: mAP graphs for Mask R-CNN and YOLOv5 models	72
Figure 6. 13: Deployed web application	73
Figure 6. 14: Deployed mobile application	74

LIST OF TABLES

Table 1. 1: Area under Coconut by Province (in Ha)	3
Table 1. 2: Comparison of former researches	12

Table 4. 1: Summary of data collection	22
Table 5. 1: Test case to verify whether the captured image is stored Google cloud storage.	58
Table 5. 2: Test case to classify and select the best model for CCI (Mask R-CNN).....	58
Table 5. 3: Test case to mask the infested leaflets	59
Table 5. 4: Test case to crop segment the masked area	59
Table 5. 5: Test case to calculate the level of progression.....	60
Table 5. 6: Test case to detect true negative results.....	60
Table 5. 7: Test case to calculate caterpillars on leaflets	61
Table 5. 8: Test case to calculate caterpillars on white paper	61
Table 5. 9: Test case to create panoramic image	62
Table 5. 10: Test case to check the integrated system	62

LIST OF ABBREVIATIONS

Abbreviation	Description
CCI	Coconut Caterpillar Infestation
CNN	Convolutional Neural Network
CRISL	Coconut Research Institute of Sri Lanka
GDP	Gross Domestic Product
CDO	Coconut Development Officers
RPN	Region proposal network
SoC	System on Chip
CSA	Channel-spatial attention
AI	Artificial Intelligence
RPN	Region Proposal Network
FCN	Fully Convoluted Neural Network
SVM	Support Vector Machine
YOLO	You only look once
CCB	Coconut Cultivation Board

1. INTRODUCTION

1.1 General Introduction

A smart agricultural solution with automated plant disease detection and classification tools has been identified as an important technique which supports efficient farm management [1]. AI integration and deep learning models are such digital tools which provide the optimum solution for the identification of plant diseases [2]. The traditional manual observation method is time-consuming and also can make errors because farmers fail to detect plant diseases in their initial stages and in large areas.

In any crop cultivation, damage due to pests and diseases accounts for a substantial reduction in the quantity and quality of yield resulting in low profitability. Coconut cultivation is no exception and according to the industry experts, inefficient management of many diseases and pests hampering the coconut production is considered as an issue that needs priority action. This industry, which is one of the most valuable sources of employment and foreign exchange, with a 0.8% contribution to the Gross Domestic Product (GDP) and 5.1% to the total export earnings in Sri Lanka needs to be protected [3]. In addition, coconut plays an important role in food security for people as it is an integral component of cuisine and a major source of fat in daily diet. The kernel of the nut is used as a component in the daily diet of Sri Lankan people providing 70% fat, 15% of calories, and 5% protein. In addition, the kernel and other parts of the nut (nut water, shell, husk) and almost all parts of the tree (trunk, leaves, ekel, flower) provides raw materials for a large number of industries. Therefore, coconut is considered a food crop, industrial crop, and livelihood crop.

The high incidence of pests and diseases is considered one of the major constraints related to coconut production and farm productivity [4]. As a perennial crop of which its productive lifespan extends up to about 50-60 coconut palm is threatened by many pests

and diseases. The stature of the palm creates difficulties to adopt traditional manual observation methods as well as pest management practices [5].

The burden placed on farming communities by heavy crop losses caused by pest and disease incidents are further worsened by the poor communication between farmer and extension personnel which make them less aware of recommended pest and disease management practices. These challenges have laid the foundation for the development of integrative mobile-based applications, which includes automated pest and disease diagnosis, disease progression level identification and dispersion pattern analysis facilitating crop management [6, 7].

1.2 Background literature

1.2.1 An overview on coconut production in Sri Lanka

According to historical records of Sri Lanka, the coconut plantations were established between Dondra and Weligama in A.D. 589. Systematic cultivation of coconuts by Europeans began in 1841 at Jaffna and Batticaloa, and at Chilaw, Puttalam and Kalpitiya area. [8]

At present coconut cultivation occupies an extent of 443,538 ha which is next to rice in Sri Lanka. The coconut industry employs approximately 135,000 people in the production and the processing sectors and contributes to the livelihood of over 700,000 growers.

Unlike the other plantation crops, coconut is predominantly a smallholder crop of which 75 % of the total land comprise from below 8 ha land holdings and the balance 25% constitutes of privately or government owned estates. Compared to the estate sector, the smallholder sector is generally unorganized and management levels are lower resulting in below average productivity, though they contribute 70% of the total annual coconut

production. Therefore, to achieve targets of increased national coconut production, advanced technologies should reach both the smallholdings and estates efficiently.

Table 1. 1: Area under Coconut by Province (in Ha)

Province	2019
North-Western	180,418
Western	64, 136
Central	22,839
Southern	52783
Northern	19,335
Eastern	16,614
North-Central	30,648
Uva	19,911
Sabaragamuwa	33,956
Total	440,638

1.2.2 Pest and diseases of coconut

Coconut is a perennial crop which has a lifespan of about 60-80 years, with a massive crown of about 30-40 leaves and bearing all-year round from the age of 5-7 years. Therefore, a palm provides an ample amount of food and shelter for the survival and reproduction of many insect pests as well as several diseases.

Coconut palms are affected by a considerable number of pests and diseases some of which reduce production or quality. The economically important pests are also known as major pests that cause fatal damage or outbreaks that result in heavy economic losses. Among the major pests of coconut in Sri Lanka coconut caterpillar (*Opisina arenosella* Walker),

red palm weevil (*Rhynchophorus ferrugineus* Olivier), Plesispa beetle (*Plesispa reichei* Chapuis) and black beetle (*Oryctes rhinoceros* Linnaeus) are indigenous pests. Coconut mite (*Aceria guerreornis* Keifer), is an introduced pest in 1998. Diseases are relatively fewer but, recently reported Weligama Coconut Leaf Wilt Disease (WCLWD) recently reported in the Southern Province is threatening the coconut industry of Sri Lanka.

Coconut caterpillar *Opisina arenosella* Walker is one of the main pests of coconut in Sri Lanka since the damage, defoliation was first reported in 1898. Thereafter frequent outbreaks were reported from coastal districts of the North-Western, Eastern, Southern and Western Provinces [9]. Research studies have confirmed that moderate to high densities of coconut caterpillar is capable of causing economic yield loss and outbreaks mostly reported between February and October. Female coconut caterpillars lay eggs on the dorsal surface of coconut leaflets. The five instars of the larval stage feed on the dorsal surface for about 30-40 days resulting in the characteristic scorched appearance of coconut leaves. The damage to coconut tree is caused by reducing the photosynthesizing leaf tissues of the coconut leaf [10].

Freshly damaged areas of the leaves appear green in colour, then gradually turn in to brown colour. The dried up patches appearing on the upper epidermis of leaves gradually extends all along the leaflets and finally the fronds develop burnt appearance. When examining the underside of the leaves, the galleries constructed with dried leaf residues, excreted material and silken web where the caterpillar hide in is visible. Infested palms could be easily recognized by the dried up patches on leaflets of the lower whorls of leaves. Though the infestations mostly occur in leaves at the lower whorls of the crown, in an outbreak, large number of leaves are affected. Figure 1.2 shows an infested coconut land and a leaf damaged by the coconut caterpillar. During severe outbreaks, the caterpillars feeds even on green surfaces of leaf petioles, spathes and nuts. Different progression levels of the infestation at initial stages and after outbreaks are illustrated in Fig 1.3

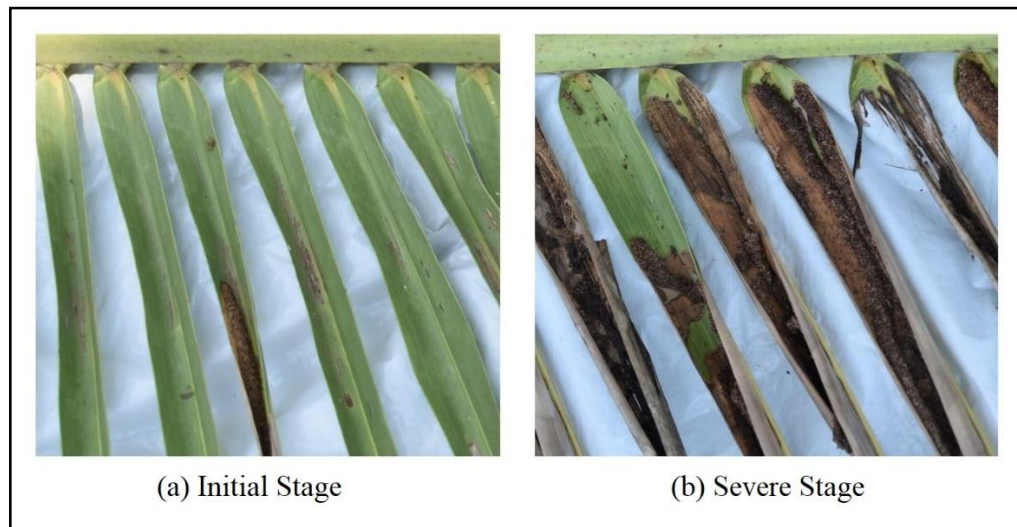


Figure 1. 1 : Different progression state

In a study conducted in Kerala, India, a maximum 45.4% loss was recorded in the nut yield from coconut caterpillar infested palms in a year after severe pest infestation. The infested palms also showed reduction in leaf and bunch production. The infested palms regained their normal yield potential about four years after a heavy pest infestation [11].

Naturally the population of coconut caterpillar is kept under control by natural enemies of the pest but the low temperature during the night time of February to October period is unfavorable for the natural enemies. Therefore, an outbreak of the pest could occur during this period. In order to prevent a coconut caterpillar outbreak which will bring an economic loss, identification of the pest at the initial stage is very important. At present chemical pesticides are banned in Sri Lanka as a government policy. Therefore, it is a challenge to manage pests and diseases at the very inception preventing possible outbreaks.

The coconut growers have been advised to notify relevant authorities, the Coconut Research Institute and the Coconut Development Officers of the area if caterpillar

infestation is identified. As the recommendation of control measures are based on the pest populations, assessment of the pest population is conducted before recommending the control measures [12].

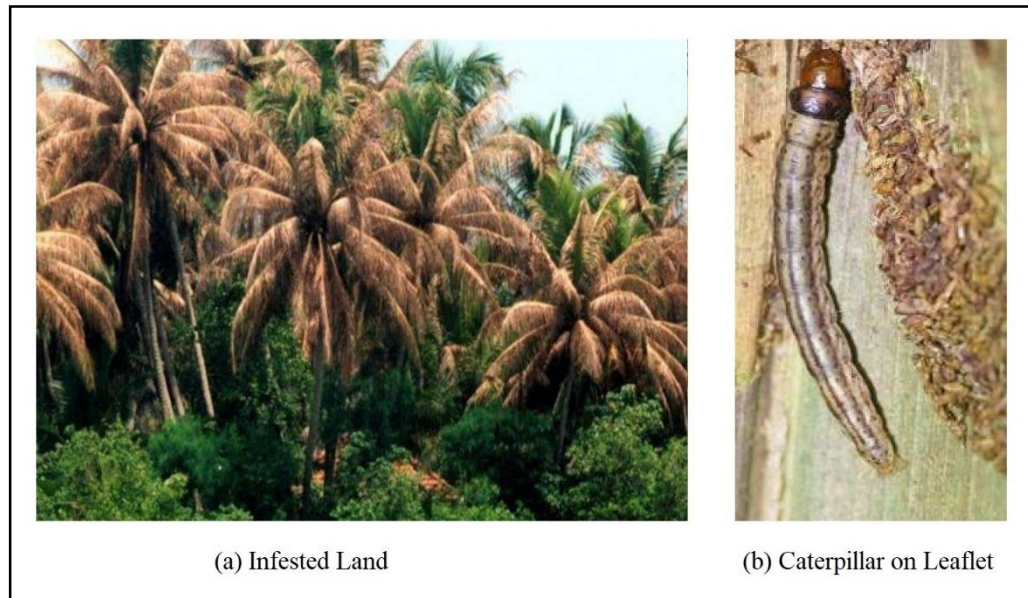


Figure 1. 2: Damage caused by Coconut Caterpillar [25]

In order to assess pest population by the manual method, 20 leaflets are randomly collected from one infested leaf of the middle whorl of an infested palm. Based on the diseased palm population, samples are collected from an adequate number of palms. The larval and pupa stages inside the galleries of leaflets are collected and counted separately. A sample data sheet is shown below in Figure 1.3. This whole process is arduous as well as time-consuming when considering the time taken for farmer observation, notification to CDO, visit of the field to collect samples, assess the samples and giving the feedback to the farmer with control measure.

කොළ රෝගී රාශි මනාප විවර්තය

මෙම පත් පත්‍රය - කුඩා පත්‍ර - 2.0
 කුඩා පත්‍ර - මධ්‍යම පත්‍ර - 3.0
 මධ්‍යම පත්‍ර - විශාල පත්‍ර - 5.0

මෙම පත්‍රය - 5

2019/12/18

මෙම පත්‍රය	කොළ අවස්ථාව			පරිසරය			මෙම පත්‍රය
	කුඩා	මධ්‍යම	විශාල	පිළිබි (පිළි)	පිළිබි (පිළි)	පිළිබි (පිළි)	
1	100	100	100	100	100	100	100
2	100	100	100	100	100	100	100
3	100	100	100	100	100	100	100
4	100	100	100	100	100	100	100
5	100	100	100	100	100	100	100
6	100	100	100	100	100	100	100
7	100	100	100	100	100	100	100
8	100	100	100	100	100	100	100
9	100	100	100	100	100	100	100
10	100	100	100	100	100	100	100
11	100	100	100	100	100	100	100
12	100	100	100	100	100	100	100
13	100	100	100	100	100	100	100
14	100	100	100	100	100	100	100
15	100	100	100	100	100	100	100
16	100	100	100	100	100	100	100
17	100	100	100	100	100	100	100
18	100	100	100	100	100	100	100
19	100	100	100	100	100	100	100
20	100	100	100	100	100	100	100

G.S.S SIVAPALLE
Coconut Development Office
Agrovet Services Center
Nabawa, Nallandana

Figure 1. 3: Sample sheet of how data is collected for coconut caterpillar infestation

1.2.3 Deep learning models for pest and disease diagnosis

Smart mobile-based applications using different artificial intelligence and neural network models were designed to control diseases and pest infestations for different crops. Prediction of wet appearance on the wheat flag leaves by environmental variables [13], identification of tea leaf diseases such as black spot, blister blight, and leaf rust using leaf colour intensity [14], detection of *Rhizopus* rot in red tomatoes using near-infrared spectroscopy [15], identifying diseases in *Phalaenopsis* seedlings using texture and colour [16] are some studies carried out using deep learning tools for disease management in agriculture. Similarly, for pest management, insect identification and control systems [17, 18], an expert system of tea pest management for appropriate control measures [19], identification of pest infected leaves using object detection techniques [20] are some of the examples reported for other crops.

According to Chandy [21] pest identification and assessment techniques that based on learning features automatically to collect more valuable information will be more sustainable in the future. The author has proposed a system for the identification of

coconut pest infestations using deep learning techniques. In this system, the pest infestations are identified by using a drone that will fly through the coconut lands in order to capture the images. It will identify the pest infestation by processing the images using deep learning and image processing. Finally, the result is sent to the user's mobile phone using WIFI [21]. Recently another study reports the detection of diseases like stem bleeding disease and leaf blight disease and red palm weevil infestation in coconut through deep learning techniques. A collection of manually collected images of both normal and abnormal coconut trees were separated using popular segmentation algorithms. First, the abnormal boundaries were marked. After marking the boundaries, pest infections and diseases were predicted using 2D-Convolutional Neural Network (CNN) [22].

Besides above-mentioned systems and techniques, instance segmentation and object detection technologies have become popular in detecting pest infestations in plants. It is used to locate the region of interest in the image precisely and to determine the specific category of each object [23]. With the introduction of Mask R-CNN, the state-of-the-art method for instance segmentation has been widely used in the domain of agriculture. Wang et. al. [23] describes use of Faster R-CNN and Mask R-CNN to identify eleven tomato diseases quickly and accurately and segment the locations and shapes of the infected areas. Mask R-CNN was used to create an automatic identification system for early detection of pests such as spider mite, whitefly, and thrips in sweet peppers [54]. Here also the deep learning model was used to determine the infection type and the area of infection. The degree of nutrition (Nitrogen, Phosphorous, and Potassium) deficiencies in crops like Guava, Groundnut and Citrus was calculated by Manoharan et. al. through calculating the unhealthy extent of the leaf using the masked images acquired by Mask R-CNN [24]. The detection and localization of non-beneficial insects of soybean was reported in order to build bounding boxes and segmentation frames for each bug in the image [25]. Application of Mask R-CNN to classify and calculate progression level of pest damages in coconut has not reported yet.

Automation of counting and classifying aphids is reported by Lins et. al. [26] In this report the authors suggest that the counting insects manually is time-consuming and prone to errors. To overcome this problem, the authors present a method and software to which enable automation of counting and classification of aphids using computer vision, image processing and machine-learning methods. In controlling coconut caterpillar damage, based on the caterpillar counts management practices are decided, but automation of counting coconut caterpillar is not reported elsewhere.

In addition to the coconut caterpillar damage, scorching or brown patches of leaves occur due to a condition known as leaf scorch disorder (LSD). The symptom of this disorder is also tip browning of the coconut leaflets at the lower whorls. Later the scorching spreads throughout the leaflets [27]. Therefore, there is a possibility of miss identification of coconut caterpillar damage as the leaf scorch disorder which is not a spreading condition. This may lead to an outbreak of coconut caterpillar due to neglect of the initial symptoms. As shown in Figure 1.4 people are not aware of the association of browning symptom with coconut caterpillar infestation (CCI) but are aware of it as scorching and nutrient deficiency symptom.

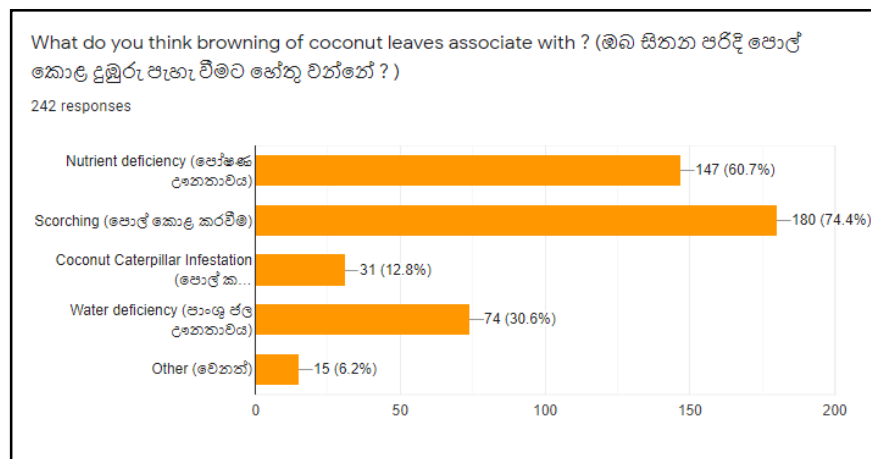


Figure 1. 4: Survey report on reasons for browning in coconut leaves

According to the research background and survey discussed above, smart surveillance of one of the major coconut pests, the coconut caterpillar to identify the infestation accurately and to calculate the progression level would benefit the coconut farming community. This will also provide necessary information for researchers to decide on the correct control measures to save the palms from damage and spreading the infestation. There are very few research reports on precision agriculture techniques to identify some pests of coconut palms in India. However, this is the first such attempt in Sri Lanka with a farmer participatory approach and the first research to provide the progression level of CCI by calculating the region of infection with the caterpillar count.

1.2 Research Gap

In coconut caterpillar infestation the damage is caused by the larvae by feeding on the lower leaf surface. According to the Coconut Research Institute of Sri Lanka (CRISL) to identify the severity of infestations, it is necessary to extract the extent of the affected leaf area as well as the number of caterpillar larvae present in that particular leaf. The research reports [28], [21] and [22] describe some studies by Indian researchers on smart agricultural solution with automated plant disease detection and classification tools such as AI integration, training deep learning models for coconut disease and pest management.

The research “A” reports on the use of image capturing drone, identification of pest infestations using deep learning techniques [21] The pest infestation is identified using a drone that flying through the coconut lands capturing the images through a GoPro camera. The drone is embedded with NVIDIA Tegra System on Chip (SoC) with ARM Cortex-based CPU and GPU for the purpose of image processing. To differentiate the pest affected and healthy areas, Region proposal network (RPN) and Channel-spatial attention (CSA) are used. The authors propose to identify the infestations but the extraction of progression level to minimize the risk of an outbreak is not considered. On the other hand, since the coconut caterpillars live on the dorsal leaf surface, the identification cannot be correctly done by examining the upper surface of the leaf. Through a drone capturing the

lower surface is difficult. The result may be misleading when coconut leaf scorching disorder is prevalent in the field. Since the drones are not affordable at farmer level, the field level operations need the expert's involvement. Therefore, pest management will be less efficient than using a mobile application approach because without the experts the farmers cannot identify the infestations.

An expert system was developed in research “B” [28], where few diseases of coconut are taken into consideration. The system only provides an overview of coconut diseases for the users. Object Expert System language called CLIPS was used in this research. The system suggests the users to select the correct answer based on the symptoms in each screen that they provide. After the discussion session is over, the proposed expert system provides the identification and recommendation of the diseases to the user. In this system, the identification process is not done using AI technologies. The identification is done only for several diseases and not for pest infestations, it depicts the diseases through the inputs of the user making the system less reliable since users lack in-depth knowledge about diseases. If the user selects the incorrect symptoms, the recommendations provided by this system will be inappropriate.

A recently published research “C” [22] have developed a system using image processing and deep learning techniques to detect leaf blight disease, stem bleeding disease, and pest infestation caused by red palm weevil. In this research a range of hand-collected images from normal and abnormal coconut tree were separated using popular segmentation algorithms. First, the abnormal boundaries were marked. After marking the boundaries with the use of a custom-designed deep 2D-Convolutional Neural Network (CNN) pest infections and diseases are predicted. The researchers have analyzed few CNN models to correctly classify and distinguish both healthy and infected trees using inductive transfer learning method. Finally, the research confirms that the most effective segmentation method was the k-means clustering segmentation. Furthermore, the custom-designed deep

2D-CNN model has achieved an accuracy rate of 96.94%. However, the detection of CCI and the progression levels are not considered in this research.

The below Table 1.2 shows a tabularized format of the above explanation with regard to classification of coconut caterpillar infestation (CCI).

Table 1. 2: Comparison of former researches

Application reference	Identification of CCI	Progression level detection of CCI		Mobile-based identification approach
		Damaged leaf area	Number of caterpillars	
Research A	✓	X	X	X
Research B	X	X	X	X
Research C	X	X	X	X
Proposed System	✓	✓	✓	✓

Since there are no reports on mobile based identification system for coconut caterpillar, in this study we propose to develop a mobile application which farmers can use to identify the infestation as well as the progression level with ease without the help of an expert. After the infestation is identified, the details of the infested palms will be sent to CRISL with the location. Based on the information, researchers can assist the farmer with the most suitable management practice or the necessary guidance to prevent further spread preventing economic loss.

2. RESEARCH PROBLEM

The productivity of any crop is reduced by pests since the beginning of farming. Therefore, protection of the crops from pests is essential to achieve high productivity levels that required to fulfill the demand for human consumption and to make agriculture profitable. The yield of cultivated crops is threatened due to destruction and competition from pests, especially when they are grown as large-scale in monocultures. Coconut is mainly grown as monoculture plantations and also it is a perennial crop of 40 to 60 years, therefore, pests can thrive well the year-round.

The leaf-eating caterpillar, a prolific feeder of coconut leaves is a serious pest of coconut palm causing significant yield loss. It infests coconut of all age groups and because of the feeding damage palm shows a scorched appearance due to the drying of leaves. Similar necrosis on the leaves also appear due to the disorder called Leaf Scorch Decline. As shown in Figure 1.5 our community is not familiar with coconut caterpillar damage and the symptoms associated with it. This can result the pest not reported to authorities at the initial stages. Therefore, educating people to identify the pest at initial stage of an infestation will pave the way to the effective management of the pest.

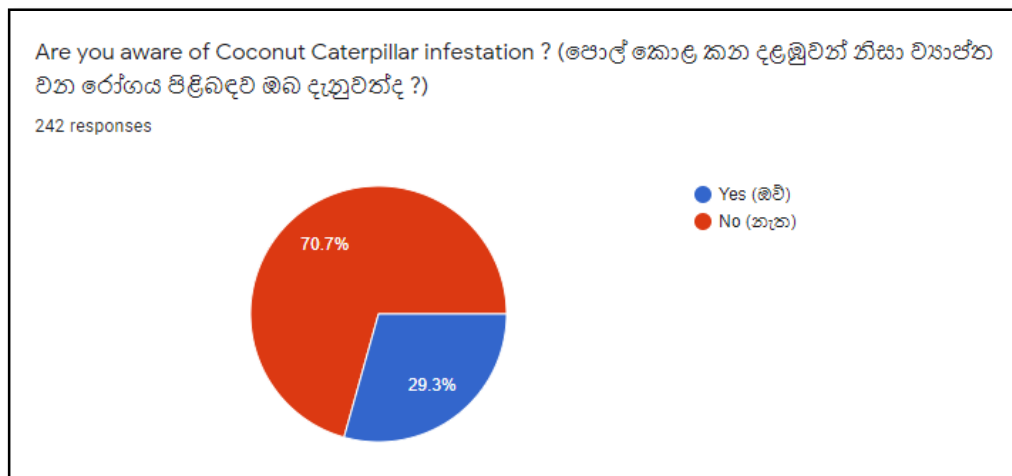


Figure 2. 1: Survey of awareness on coconut caterpillar infestation

The CRISL has developed an efficient biological control method for coconut caterpillars by identifying, breeding and releasing a natural enemy of the pest. With the increase of severity, the biological control method needs to be supported with chemical control of the pest [12]. Therefore, identifying the severity of the disease by the community itself will speed up the selection of specific management practices. According to the survey, many are willing to get the knowledge on deciding the severity percentage as shown in Figure 1.6.

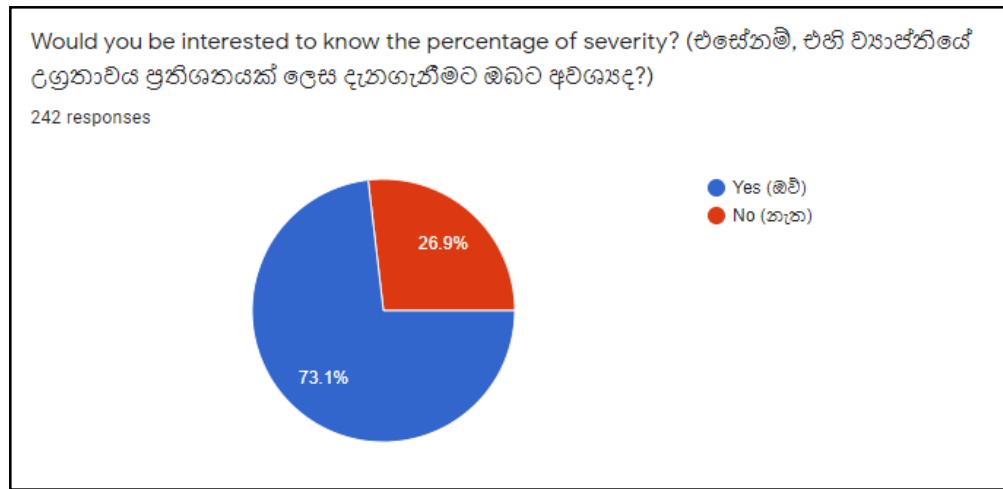


Figure 2. 2: Summary of willingness to learn about the severity

There are no reports on smart agriculture tools developed for the identification and determination of the severity of coconut caterpillars by the coconut growers or the community. Therefore, the present study proposes a mobile application system where the stakeholders can easily identify the pest differentiating the symptoms from leaf scorch disorder and report the severity of the infestation to relevant authorities to speed up the control of the pest avoiding possible outbreaks which can result in an economical yield loss.

3. RESEARCH OBJECTIVES

3.1 Main Objectives

The main objective of this study is to develop a mobile application through which coconut growers can easily identify the coconut caterpillar infestation while differentiating its symptoms from other conditions with similar symptoms (leaf scorch disorder). A photograph image of an infested leaflet will be the input given by the coconut grower. The severity of the infestation will also be determined at the same time, and appropriate authorities will be notified in order to expedite the application of control measures to prevent possible outbreak of the pest.

3.2 Specific Objectives

There are three specific objectives that need to be fulfilled in order to achieve the overall objective described above.

Identify Coconut Caterpillar Infestation (CCI)

- Photograph of a coconut leaflet will be analyzed using Deep Learning techniques such as Faster R-CNN which is a part of Mask R-CNN to classify browning of leaves, availability of galleries (Larvae construct galleries of silk and debris, in which they hide when disturbed) and caterpillars to identify whether the coconut leaf is infested with coconut caterpillars.

Identifying the progression level of infestation

- To measure the extent of browning throughout the coconut leaf and to mask the progression level of damage, Mask R-CNN is used to accurately segment the area of the infested leaflet. After masking, the leaflet will be extracted from the background (using crop segmentation). Finally, K-means clustering algorithm is

used to calculate the percentage of infection throughout the leaflet which will help the researchers to decide on control measures.

Extracting the number of coconut caterpillars

- To identify the severity and the active state of infestation, total number of caterpillars will be counted using computer vision (OpenCV) and object detection algorithms (YOLOv5) for the researchers to decide on a control method. All the difficulties and human errors caused in the manual method is minimized by this solution.

4. METHODOLOGY

4.1 Materials and methods

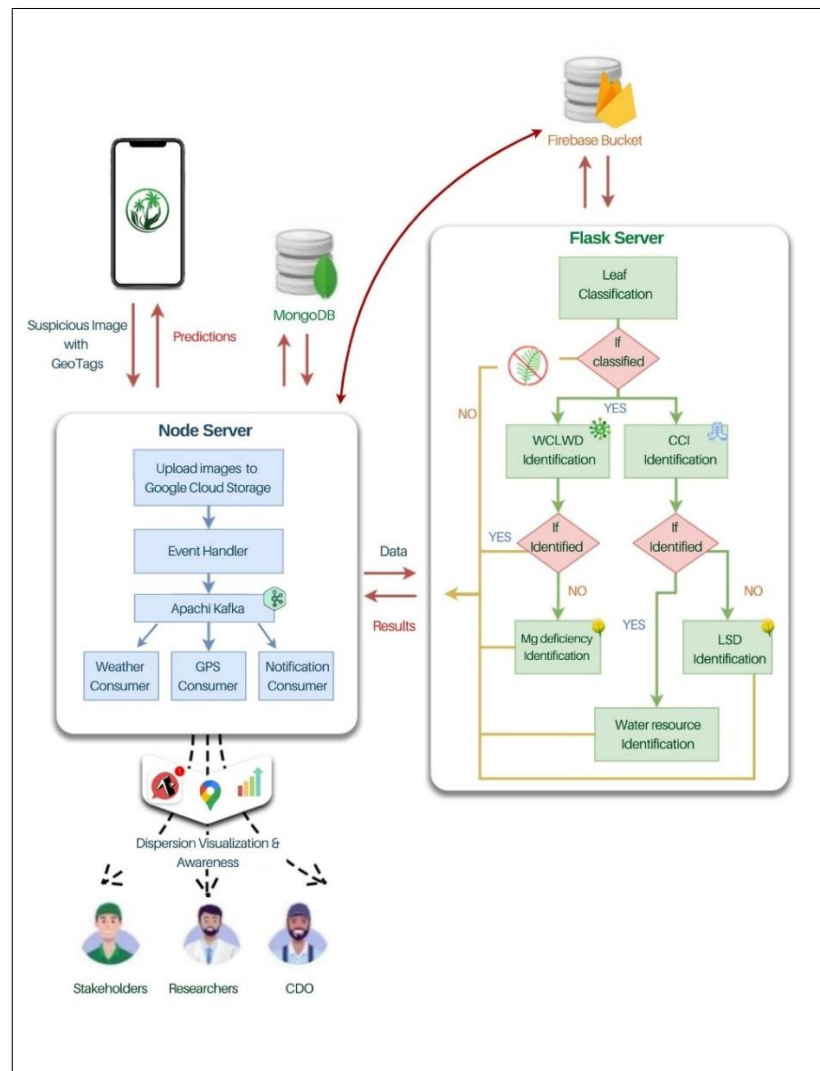


Figure 4. 1: Overview system diagram

Fig 4.1 illustrates the overall system diagram of the proposed solution which was intended to provide a smart approach for stakeholders, researchers, and Coconut Development Officers (CDOs) to detect the coconut diseases and pest infestations that may affect the coconut industry. As shown in the diagram, the registered users of the system can capture or upload the images which are suspicious. The images are sent to the Amazon Web

Services (AWS) backend server where the flask server is deployed. These images are processed in the flask server by the designed DCNN models for disease identification. If WCLWD is found, the symptom severity will be determined using CNN models. Simultaneously, if CCI is identified, the images will go through the Mask-R-CNN model to determine the progression level while the number of caterpillars are extracted using the YOLOv5 object detection algorithm. Images will be classified using the CNN models of Mg deficiency and LSD at the same time.

Once the system identifies that the leaves are infected, then the response will be captured by the crowdsourcing platform. The Google Map will be updated with the real time locations (latitude and longitude) of the infected palms. In Addition, the system will automatically send notifications to the farmers and other stakeholders who are at the risk of infection.

The overall system was divided into four main components.

1. Identification of CCI and the calculation of progression level
2. Identification WCLWD and its symptom severity
3. Identification Mg deficiency and Leaf Scorch Decline
4. Crowdsourcing and visualization of dispersion.

4.1.1 Problem statement

This research component discusses the solution to the problem of classifying coconut caterpillar-infested with non-infested coconut leaflets, as well as the calculation of the progression level. Progression level is calculated by comparing the ratio between the areas of green and brown (necrotic leaf area due to feeding of caterpillars) pigments in the leaflets. This will allow authorities to determine the stage of infestation and recommend appropriate control measures to control the situation. Furthermore, the number of coconut caterpillars available on the sample leaf is extracted to determine the active state of the infestation as well as to determine the amount of pesticides needed to control the outbreak.

The main objective of this research is to make awareness to farmers and stakeholders regarding the damage and severity of CCI that can occur in their lands.

4.1.2 Component System Architecture (Solution Design)

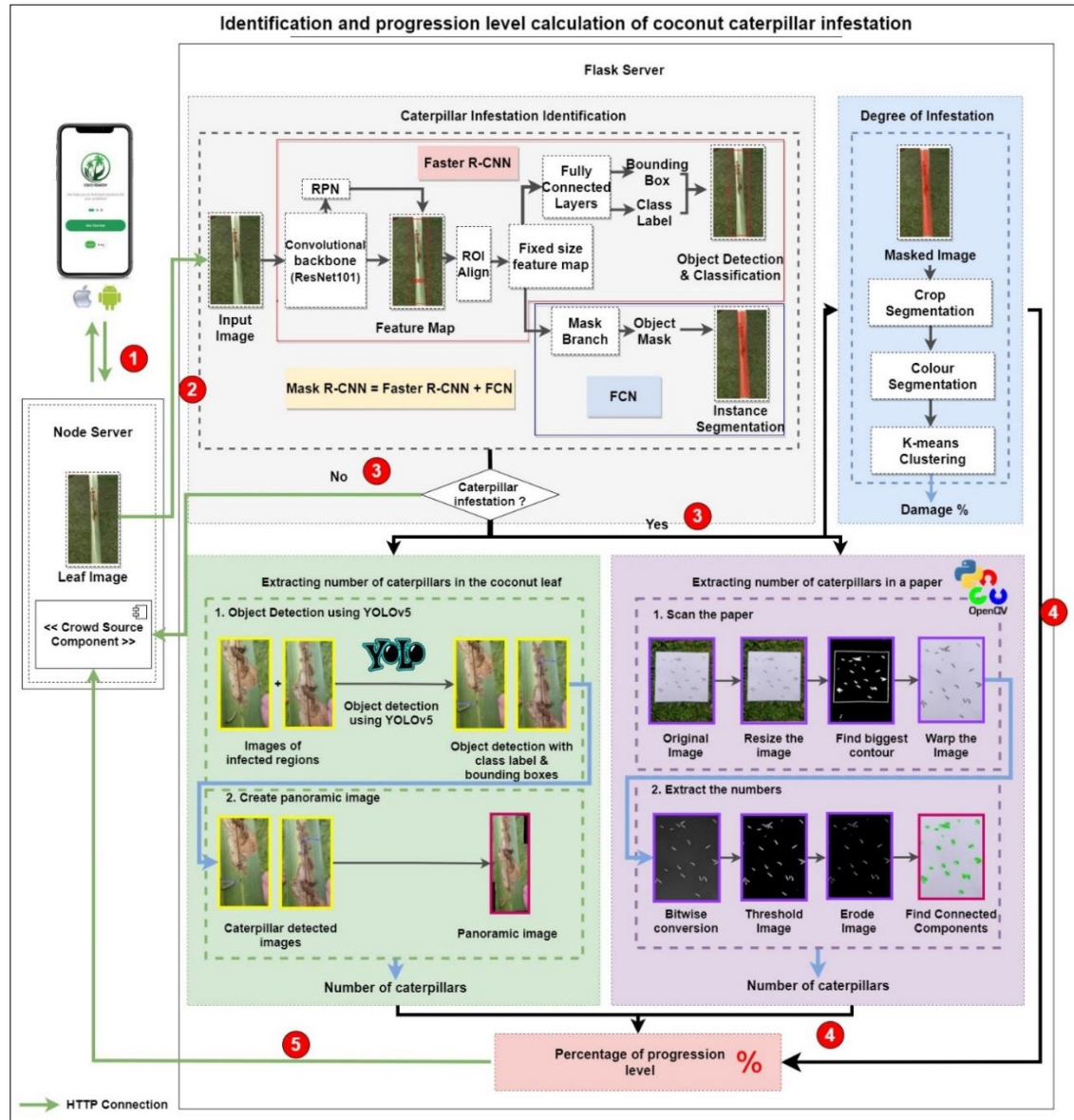


Figure 4. 2: Overview of component diagram

Fig. 4.2 illustrates the overall high-level architectural diagram of the proposed component on identifying, classifying and calculating the progression level of coconut caterpillar infestation. Initially, the stakeholder is provided with the opportunity of capturing an image or uploading a previously taken image to the system using a cross platform mobile application implemented using React Native. The captured images and GPS location data are sent as "multipart/form-data" to the backend node server. The images are uploaded to Google Cloud Storage using multer; middleware for managing form data. Finally, GPS location data, together with the file names, are sent to the Flask server to determine the coconut caterpillar infestation using deep learning technologies.

There are three main components (objectives), where first the system determines whether the leaf is infested by the coconut caterpillar. If not, it generates a message and displays that the inserted image is not of an infested leaf. The infested leaf is then extracted from the background in order to calculate the severity of infestation. Finally, the total number of caterpillars will be counted in order for the researchers to decide on a control method. The number of caterpillars are calculated either from a white paper (supports manual method) or from the leaflet itself to ease the process for the users.

In order to achieve the above objectives, deep learning models were trained using images of both infested and non-infested coconut leaves. The images were collected from coconut estates where the disease is prevalent. The quality of data (images) should be evaluated prior implementing machine learning or deep learning algorithms. Therefore, the images were processed accordingly.

4.1.3 Data Acquisition and processing

Experimental images were acquired from several estates in Lunuwila, Makandura and Puttalam in the coconut triangle where the caterpillar damage is prevalent. Deep Learning (DL) models were trained using these images of newly collected coconut leaflets. Images were captured in a variety of natural settings to avoid image sample similarities. Leaf

sampling was carried out with the assistance of trained personnel from the Coconut Research Institute of Sri Lanka (CRISL). The collected raw data is preprocessed and augmented prior training the model. Fig. 4.3 illustrates collected sample images of coconut caterpillar infested coconut leaflets at various stages of progression along with non-infested leaflets.

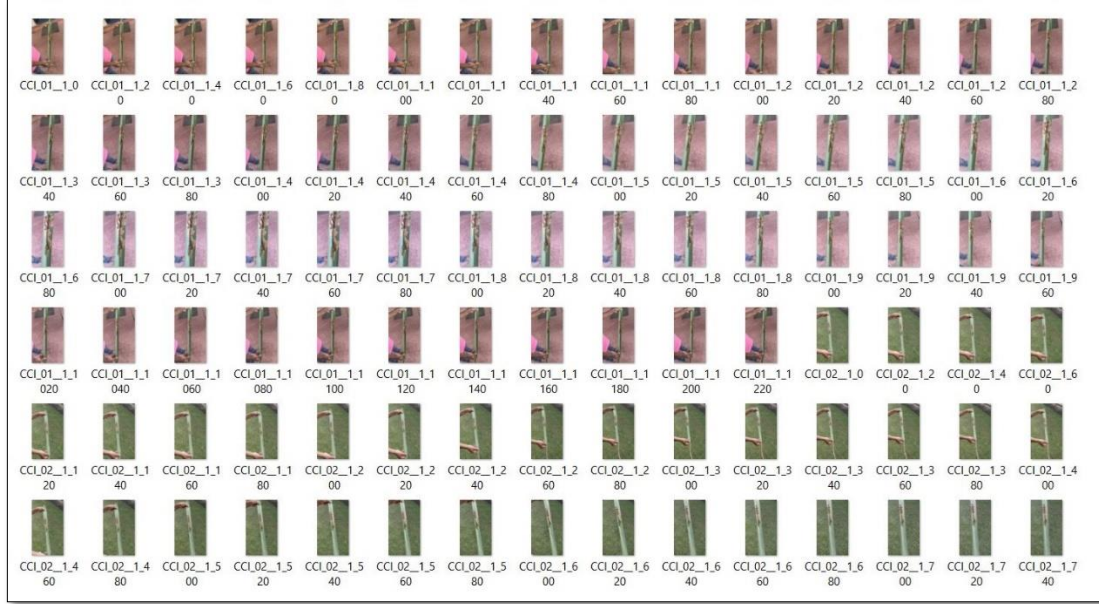


Figure 4. 3: Collected sample images used to train the model

Preprocessing and Augmentation

Preprocessing techniques were used to improve accuracy and reduce dataset complexity, which contributes in decreasing computation and running time to train the model. Since the collected images are of different sizes, they were resized into the same dimensions. Data augmentation techniques such as rotation, fill, shear (horizontal and vertical), flip (horizontal and vertical), and zoom were applied using Roboflow (open-source computer vision developer framework) as indicated in [29] to increase the number of datasets and to avoid overfitting of models. Finally, the images were divided into training (80%) and

validation (20%). After training, a new set of images were collected to test the model. Table 4.1 summarizes the statistics of data collected and data after augmentation.

Table 4. 1: Summary of data collection

Condition	Location	Purpose	Type	Number of Images			
				Total	Train	Val	Test
CCI Infected + Not Infected	Puttalam, Makandura, Lunuwila	Infestation Classification	Collected	1600	1280	320	160
			Augmented	3200	2560	640	160
		Caterpillar Calculation	Collected	1400	1120	280	140
			Augmented	2800	2240	560	140

4.1.4 Infestation identification and classification

For the identification of coconut caterpillar damage, a custom Mask R-CNN model was trained. Initially, to training the custom Mask R-CNN model, all preprocessed and augmented images were annotated manually (Fig. 4.4). Image annotation is a labeling process in which objects of interest in images are labeled with attached metadata so that they can be learned and recognized when the model is trained. Leaf images with and without the infestation damage was labelled and the remaining region was set to the background. Finally, the annotated datasets are used to train the model to appropriately identify data or predict the outcomes.

VGG Image Annotator (version 2.0.11) was used to annotate the dataset in order to generate masked images of coconut leaflets. The labeling technique used was polygon annotation. It is a simple annotation tool that can create a single JavaScript Object Notation (json) file. The file contains the coordinates of the coordinates and information for the polygonal regions. To train the model, these defined regions are fed into convolutional neural networks as input neurons.



Figure 4. 4: Data annotation using VGG annotator

Faster R-CNN which is a part of Mask R-CNN is used to identify the infestation using object detection. There are two main stages of Faster R-CNN. First, a Region Proposal Network (RPN) which uses a Convolutional Neural Network (CNN) architecture to propose the regions which recommends candidate object bounding boxes. The next step, which is more likely similar to the Fast R-CNN architecture, extracts the features of each candidate box utilizing RoIPool, conducts classification to identify the pest infestation, and uses bounding-box regression to mark the region of the infested leaf with a label [30].

Resnet101 was used as the backbone CNN network for the classification, with a combination of Feature Pyramid Network (FPN) architecture for feature extraction due to its effectiveness compared to other models [31, 29, 32, 30]. The ResNet101 architecture is divided into five stages, each of which corresponds to a different scale of feature maps. These feature maps are used to create the feature pyramid of FPN network.

Convolutional Neural Networks (CNNs) are a type of Deep Neural Network (DNN) that have been shown to perform well in image classification and object recognition. CNN is composed of three layers convolutional, pooling and fully connected layer.

- Convolutional Layer is the first layer that is used to extract various activation features from images. This layer transforms an input image in order to extract features from it. In this transformation, the image is convolved with a filter, also known as a kernel. [33]. A kernel is a small matrix with dimensions less than the input image that needs to be transformed. It is also known as a convolution matrix or a convolution mask. The dot product between the kernel and the sections of the input image with regard to the size of the kernel is taken by sliding the filter across the input image. A stride is the sliding size of the kernel. [33]. The output image is known as convolved feature map, and it contains information about the image such as its edges, colours and corners. The convolutional operation is shown in Fig 4.5.

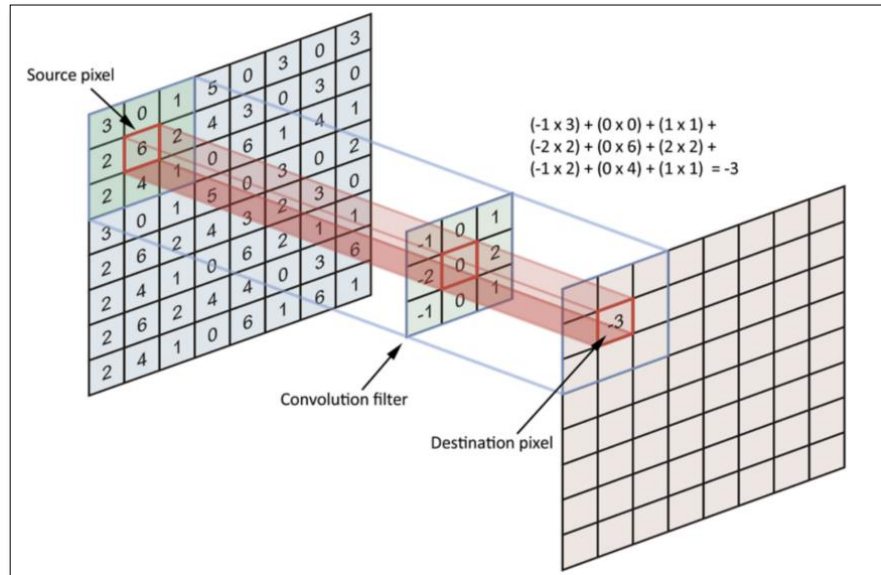


Figure 4. 5: Convolutional operation [33].

- Convolutional layer is followed by a pooling layer which helps to reduce the computational cost by decreasing the size of the convolved feature maps. It also speeds up computation and improves the stability of several observed characteristics. [33]. There are different types of pooling. Max pooling and average pooling are the two most used approaches for pooling a convolutional neural network. To generate a reduced map, max pooling selects the maximum value from each area of a feature map. However, average pooling selects the average value from each area of a feature map [33].

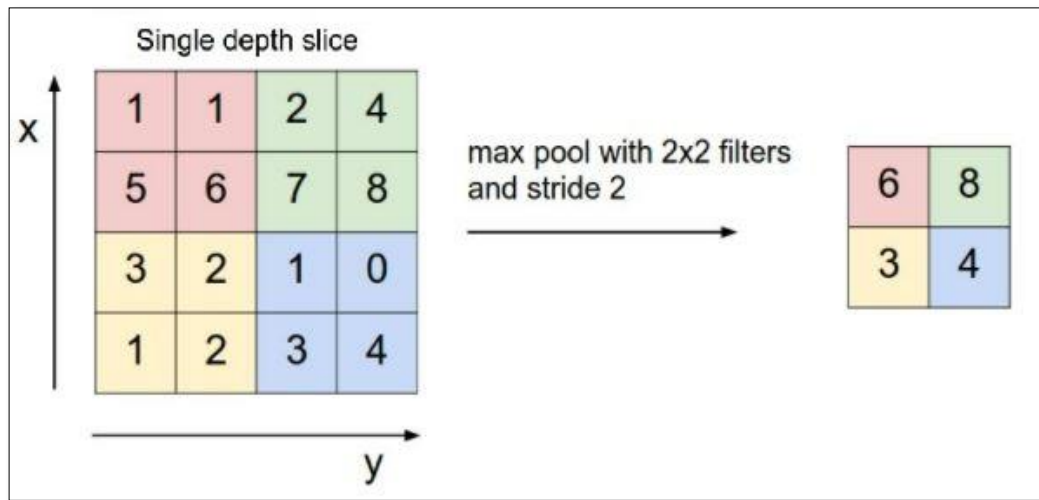


Figure 4. 6: Pooling operation [33]

- A fully connected (FC) layer is present at the end of a convolutional neural network. In the FC layer, the feature maps created in the previous layers are reduced into a vector to detect complex interactions between features at the highest level. This layer produces the one-dimensional feature vector.

The feature maps returned from the backbone CNN were sent to the Regional Proposal Network (RPN) which scans the feature maps using a sliding window to locate regions of interest (RoIs), where the leaflets exist (Fig 4.2). The RoIs are marked based on the values

of Intersection over Union (IoU) [34]. IOU refers to the ratio between the intersection and union of the ground truth bounding box and the bounding box output by the model (2). RoIs are considered if the IoU of each estimated region is greater than or equal to 0.5. The remaining regions are set to be the background.

$$IoU(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (1)$$

Where,

A = Ground truth bounding box

B = Predicted bounding box

$|A \cap B|$ = Area of overlap (intersection)

$|A \cup B|$ = Area of union

Fig. 4.7 shows an example visualization of IoU values using ground-truth bounding boxes (green) and predicted bounding boxes (red).

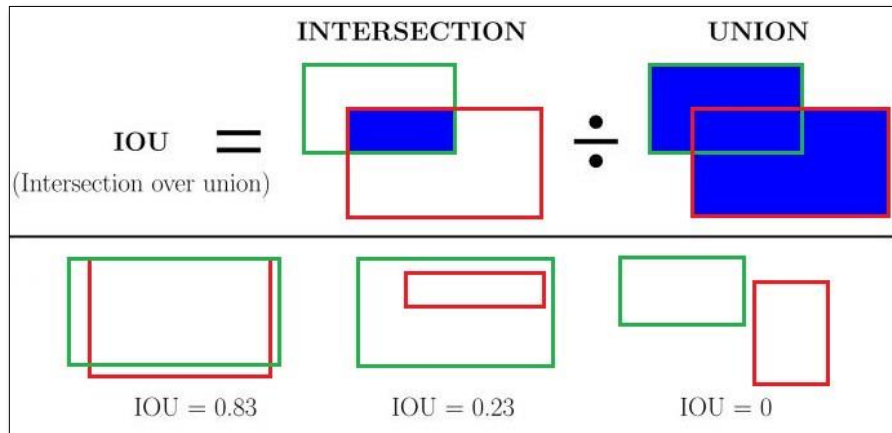


Figure 4. 7: An example of computing IoUs for various bounding boxes [35].

Each RoI region acquired on the image is represented by a bounding box rectangle known as an anchor. If several bounding boxes overlap, Non-Maximum Suppression (NMS) [25]

is used to select the bounding box with the highest foreground value. After that, RoIAlign is used to adjust the dimensions of the anchor boxes in order to create a standard-sized output (Fig 4.8). This helps to resolve the misalignment between the feature map and the source image by reducing the quantization loss. The backpropagation algorithm (2) of the RoIAlign layer is used to tune the regional proposal network [32].

$$\frac{\partial L}{\partial x_i} = \sum_r \sum_j [d(i, i \times (r, j)) < 1] (1 - \Delta k) (1 - \Delta w) \frac{\partial l}{\partial y_{rj}} \quad (2)$$

Where,

d = distance between two points.

Δk and Δw = difference between x_i and $x_i \times (r, j)$

$i \times (r, j)$ = coordinate position of a floating point number

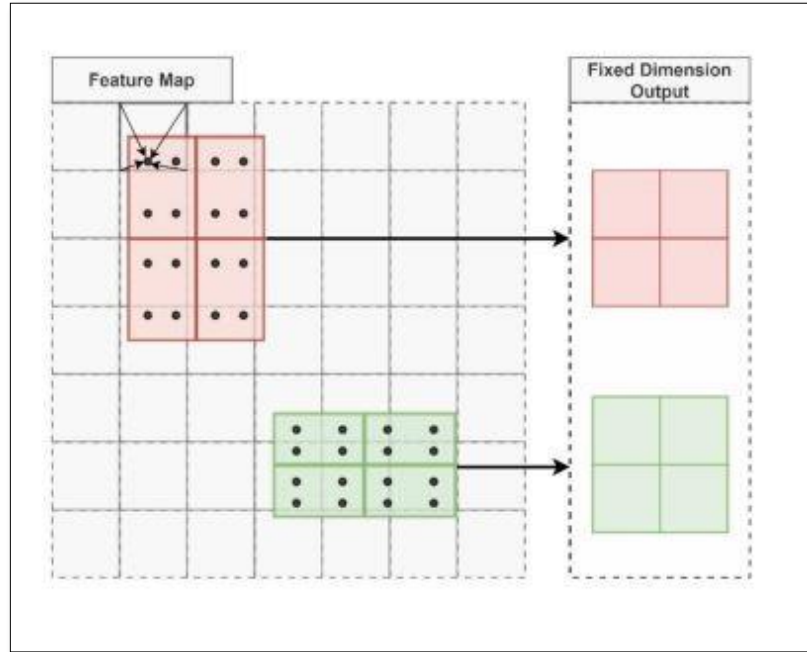


Figure 4. 8: Region of Interest align (RoIAlign) method [36]

Finally, the features obtained by RoIAlign are used to perform target classification and bounding-box regression in the FC layer. Through this process healthy and infested leaflets were classified.

4.1.5 Identification of the Degree of Diseased Condition in Leaves

The control measures applied to CCI differ with the progression level and the number of caterpillars available. Progression level is the extent of infestation throughout the leaflet. In parallel with the existing branch for classification and bounding box regression in Mask R-CNN which was used to classify CCI A, pixel-level instance segmentation performed in the Fully Convoluted Neural Network (FCN) was used to mask the infested leaflets (Fig. 4.9). The model performance was assessed by comparing the annotated mask images with the prediction results of the mask. After applying the mask, crop segmentation was used to extract the infested leaflets from the background. Finally, colour segmentation was applied to calculate the ratio between green and brown pigments caused by the damage using K-means clustering algorithm (Fig 4.9).

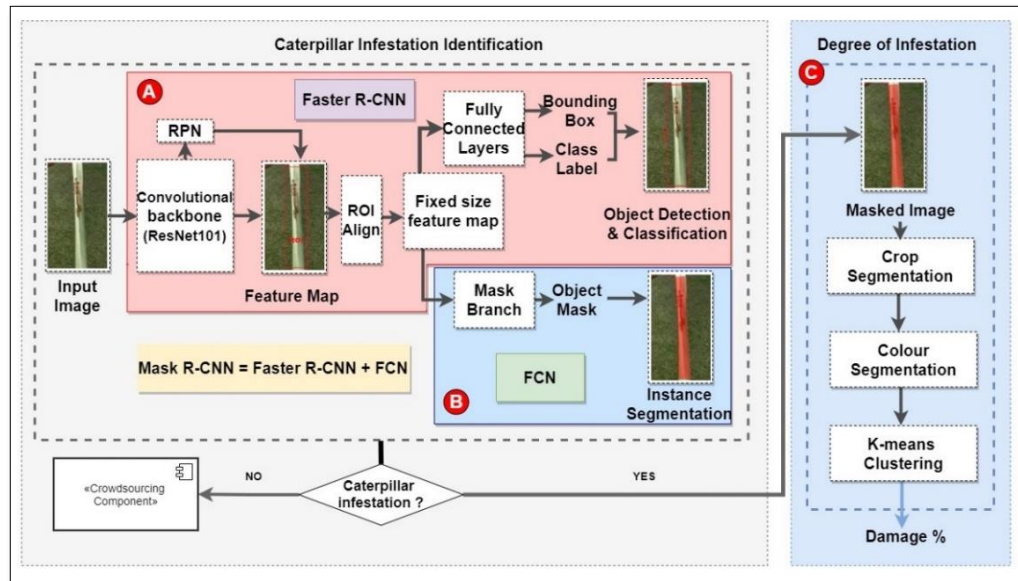


Figure 4. 9: Mask R-CNN framework with damage degree computation

In Mask R-CNN, an additional FCN layer (B) is applied to each RoI predicting a segmentation mask for the identified object (CCI infested leaflet) on a pixel-to-pixel basis. This provides us with a far more detailed understanding of the object(s) in the image. There are two types of image segmentation techniques: semantic and instance segmentation (Fig 4.10).

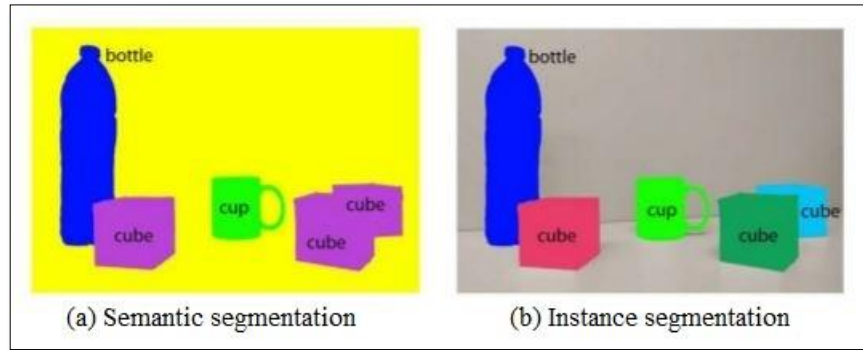


Figure 4. 10: Image segmentation techniques[37]

Semantic segmentation (a) is the pixel-labeling process that assigns a class label to each pixel of an image. It does not differentiate between objects of the same class (multiple objects of the same class are treated as a single entity) [32]. Whereas, instance segmentation (b) identifies each object instance by using a mask representation in the image, which does both object class prediction and mask extraction at the same time [32]. It provides distinct labels for separate occurrences of objects belonging to the same class. Instance segmentation consists of three steps: Identifying proposal regions with RPN, predicting object class, and mask extraction. In this research, instance segmentation is considered with the intention of differentiating each instances of CCI identification.

The leaf area of the image is masked, and only the class label defined as CCI is passed to the progression level determination component (C) If the image is not identified as CCI, the result is passed to the crowd sourcing component notifying the users (Fig: 4.9). Using

the mask coordination and dimensions, the infected leaf is only extracted while the background is separated. After crop segmentation the ratio between green and brown (necrotic leaf regions due to feeding of caterpillars) areas are calculated while eliminating the white background. Using colour segmentation, range of HSV colours that belongs to green, white and brown was replaced with a single occurrence (pixel values) before using K-means clustering algorithm to calculate the ratio between healthy and damaged parts of the leaflet.

K-means is a type of centroid-based clustering algorithm which is used to group objects with similar features together. A centroid is a data point in the center of a cluster. Centroid-based clustering is an adaptive technique that compares data points depending on how close they are to the cluster's centroid. It attempts to make intra-cluster data points as similar as possible while maintaining clusters which are different as far as possible. In K-means, x data points are subdivided into a set of k clusters with each data point allocated to the nearest cluster. This process is defined by the objective function (3) that minimizes the sum of all squared distances within a cluster for all clusters [38]. The objective function is defined as:

$$J = \sum_{j=1}^k \sum_{i=1}^n \|x_i^{(j)} - c_j\|^2 \quad (3)$$

Where,

$x_i^{(j)}$ = Data point

c_j = Center point of clusters

n = Number of data points

k = Number of cluster

$\|x_i^{(j)} - c_j\|^2$ = Distance between $x_i^{(j)}$ data point and c_j cluster centre.

Fig 4.11 depicts a step-by-step representation of the clustering process of x data points into three clusters (k). Similarly, all of the green (healthy), white (background) and brown (damaged) pixel values in the extracted leaflet caused by the infestation were clustered independently in order to calculate the CCI damaged percentage.

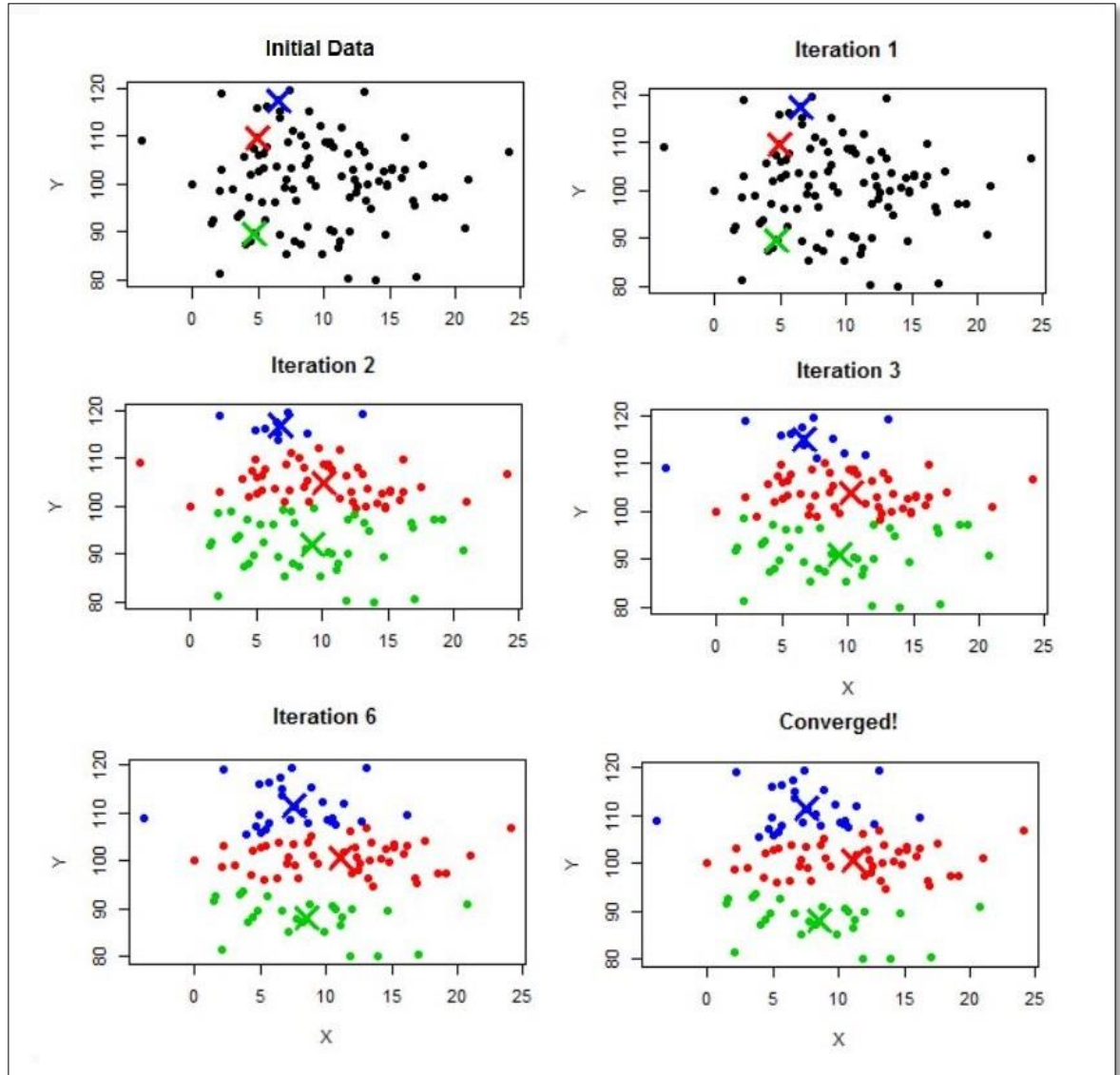


Figure 4. 11: K-means clustering process [39]

4.1.6 Identification and counting caterpillars

At present, counting is done by manual collection of caterpillars on a paper. This process is time consuming and prone to human error. The existing method was automated by image processing techniques to make it efficient, accurate and less time consuming. In addition, a state of art advanced technological method was developed to count the caterpillars using You Only Look Once (YOLO) object detection algorithm. This method facilitates counting of caterpillars at their natural habitat (leaflet itself) without forcefully extracting on to a paper.

Caterpillar calculation using image processing

- **Scanning the Paper**

Agriculture practitioners have found it challenging to count caterpillars manually. Each caterpillar on the leaf sample is taken out individually and placed on a sheet of white paper. Finally, the number of caterpillars is counted one by one. Since the caterpillars move, calculating has become a tedious task. Through this automated process, the user only needs to capture an image of the paper to count the caterpillars, saving time as well as making the counting process more accurate. Initially the system scans and crop the white paper out of the image by removing the background. This process includes series of image processing steps computed using TensorFlow and OpenCV libraries. Fig. 4.12 depicts an overview graphical representation of the main stages used to scan the paper.

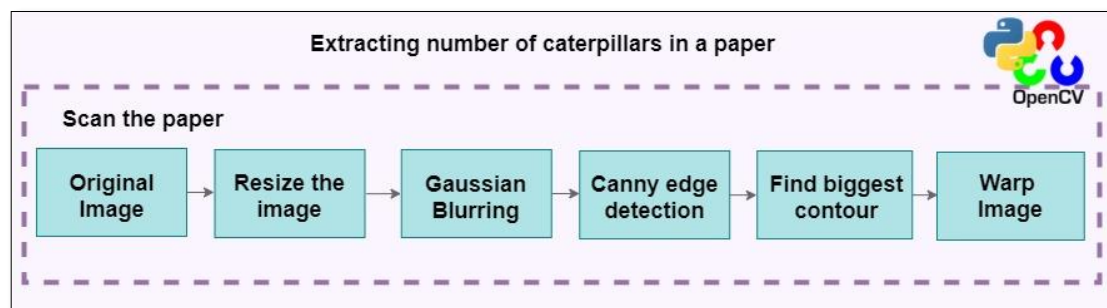


Figure 4. 12: Overview process of scanning the white paper using image processing

Initially, GaussianBlur (image blurring technique) was applied after converting the resized image into greyscale (colours in the shades of grey). It is a type of blurring filter that implements a Gaussian function to calculate the transformation to be applied to each pixel in the image. The image is blurred by convolving it with a reduced filter kernel. It is effective for eliminating background noises. When this filter is applied, it removes high frequency information such as noise and edges from the image, resulting in blurred edges.

The formula for a Gaussian function [40] in a two-dimensional image is defined as:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}} \quad (4)$$

Where,

$G(x, y)$ = Gaussian function for x and y coordinates (2D)

x = Distance from origin in horizontal axis

y = Distance from origin in horizontal axis

σ = Standard deviation

A convolutional matrix is formed from the distribution values and applied to the image. Each pixel's new value is set to a weighted average of that neighboring pixel. The value of the original pixel receives the highest weight (since it has the largest Gaussian value), and nearby pixels receive lower weights as their distance from the original pixel rises. This produces a blur that maintains edges and boundaries better than other blurring filters [41]. The visual illustration of the convolution process on a rice image [42] is shown in Fig 4.13.

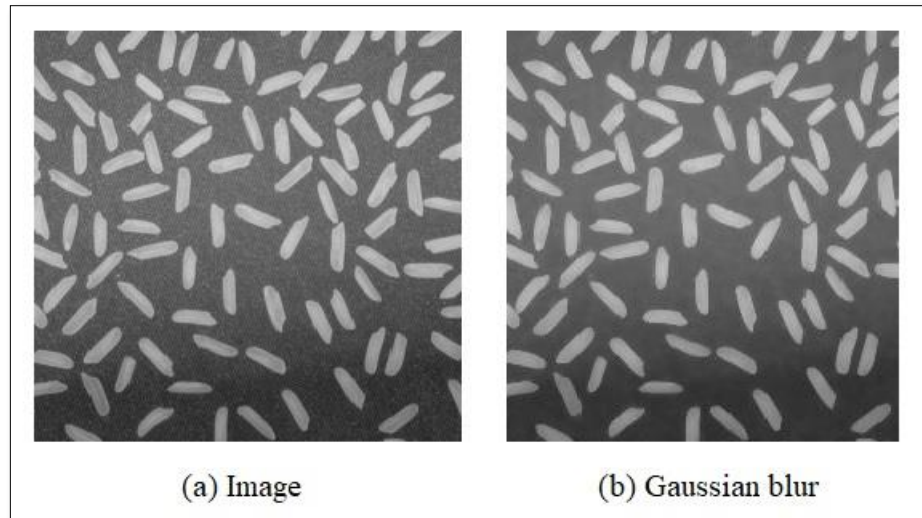


Figure 4. 13: Applying Gaussian blur to a noisy image (rice)

The Canny edge detection algorithm is applied to the blurred image to detect the potential edges (d). It is generally acknowledged as the most widely used edge detector in computer vision and image processing [41]. The boundaries are detected using canny edge detection as shown in Fig 4.14.

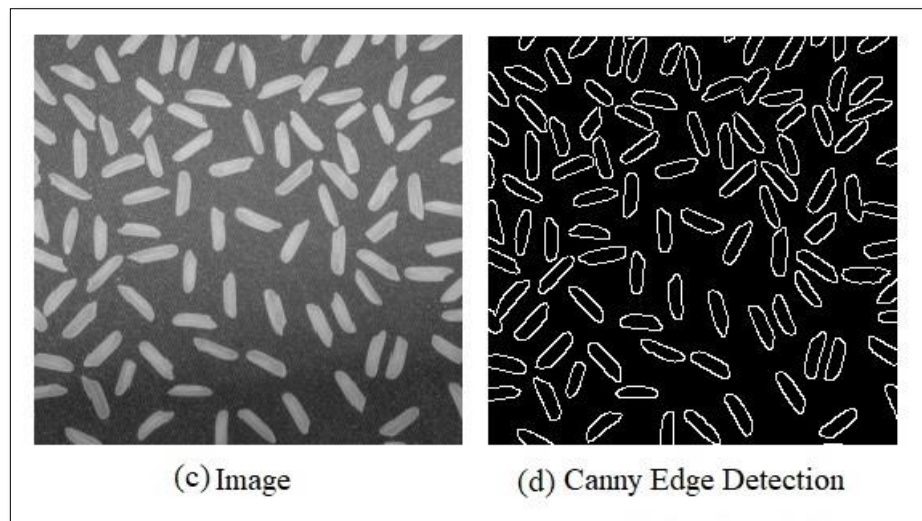


Figure 4. 14: Applying Canny edge detection

Finally, the largest contour line is identified using contour detection (which will be the outline of the white paper). A contour is a complete line that connects all continuous points that have some colour or intensity which represent the forms of objects in a picture. After identifying the largest contour, the image is warped to save only the area within the white paper. To eliminate any fault detections, it is critical to scan and extract the white paper. Otherwise, certain background objects may be counted as well.

The number of caterpillars are calculated after extracting the area under the white paper. Fig. 4.15 depicts the remaining steps involved in the detection and calculation process.

- **Calculate caterpillars**

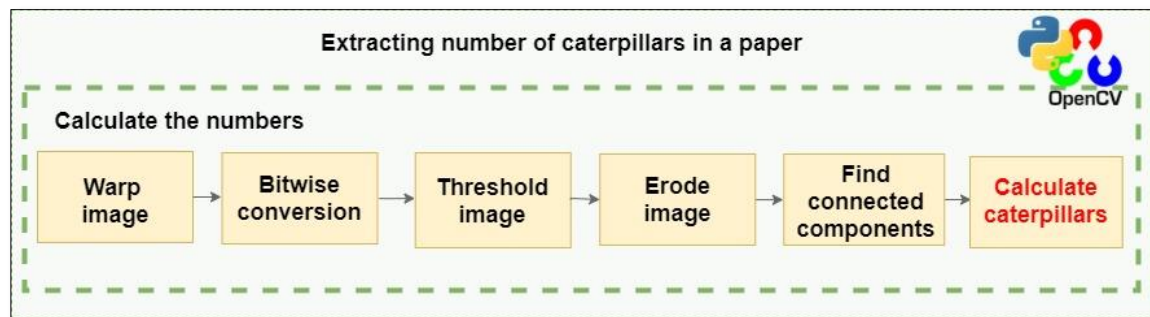


Figure 4. 15: Overview process of calculating caterpillars using image processing

The scanned image is converted to a binary image using thresholding (e) after flipping the pixel values using bitwise conversion (to clearly see the objects available). Thresholding transforms each pixel in a picture with a black pixel if the intensity value is less than a certain fixed constant, and with a white pixel if the intensity value is higher than that constant (the number is stored as an 8-bit integer where 0 is black and 255 is white.). All the dark regions are turned black while the light regions are turned white. This is used to prevent wide range of colours to be present at the image.

Erosion is a morphological operation that reduces the features of an image (shrinking the foreground). Morphological transformations are fundamental procedures that are dependent on the structure of an image. It is typically applied to binary images (threshold) [43]. It is capable of removing minor noises from images. Erosion can eliminate dust and soil particles that came with the caterpillars. It is also used to separate and identify caterpillars that are closely attached together. Fig 4.16 demonstrates a sample visualization of eroding used to separate closely attached rice and to remove excess noises (f).

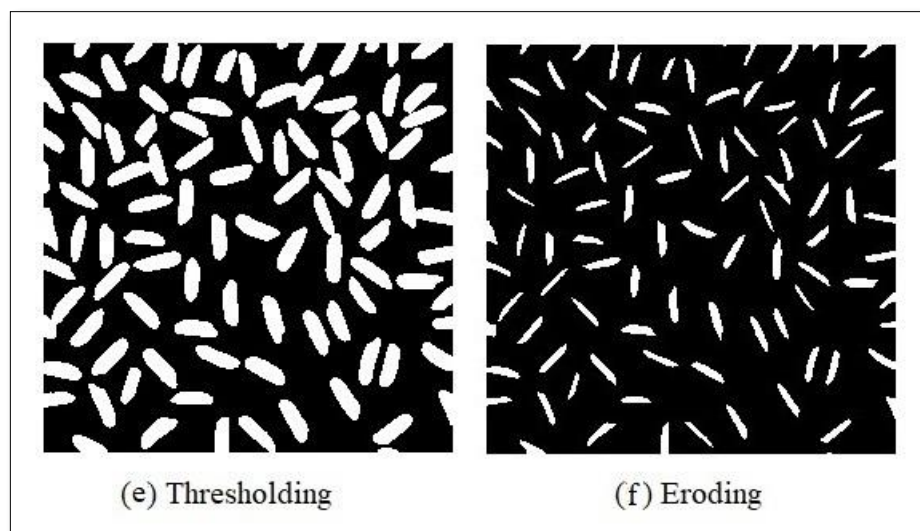


Figure 4. 16: Applying thresholding and eroding

Connected components are detected which are defined as neighboring pixel regions with the same input value [44]. Methodically the algorithm looks for white pixels, assigning each pixel with a unique label id and the same label to all neighboring white pixels iteratively until the entire caterpillar is covered and labeled. Then it moves on to the next accessible white pixel and repeats the process until the entire image has been analyzed. Finally, all the connected components are marked and calculated to detect the number of caterpillars available in the system.

Caterpillar calculation using object detection

As an innovative solution, counting caterpillars on the leaf without extracting on to a paper is attempted as it will be easier and even a farmer can pass the information without waiting for experts. Object detection was used to provide this solution where caterpillars in the leaflets were identified and localized. The process of recognizing the location and drawing a bounding box around the object is referred to as object localization [45].

The number of caterpillars are extracted using the You Look Only Once (YOLOv5) version 5 object detection algorithm. It is a state-of-art model that uses a single CNN for both object classification and localization. YOLO is built on the Darknet architecture, which consists of 24 convolutional layers followed by 2 fully connected layers. A custom model with coconut caterpillar class was used for training. After identifying and counting the caterpillars, images are stitched together creating a panoramic image to be displayed for the users. Fig. 4.17 depicts the overall process for this sub-component.

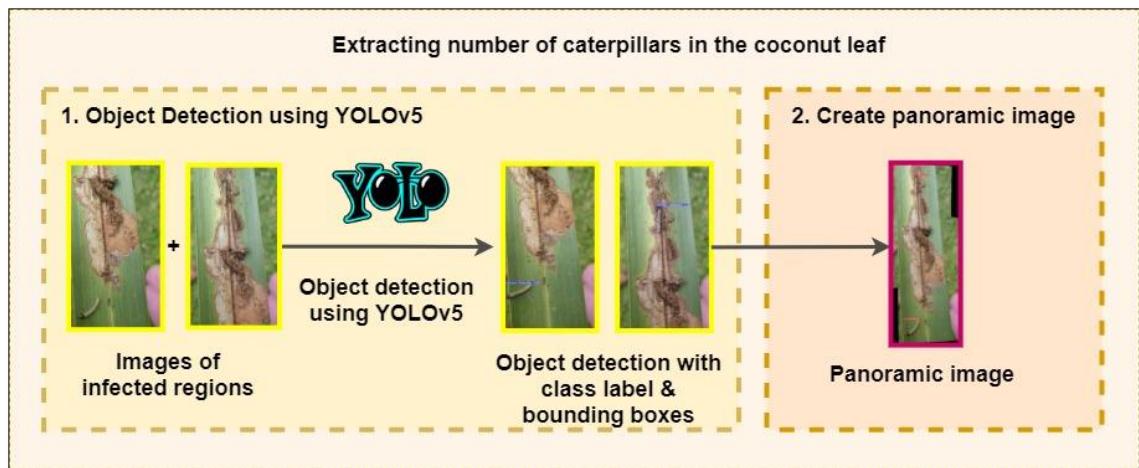


Figure 4. 17: Overview process of calculating caterpillars using image processing

Unlike Mask R-CNN, the images used to train the YOLOv5 model was annotated using Makesense.ai (online) open-source tool [46]. The labeling technique used was box

annotation. Makesense.ai also generates a single JavaScript Object Notation (json) file. The json file holds to coordinates of the box regions and information of these regions. These defined regions are sent to the CNN (DarkNet) backbone of YOLOv5 model as input neuron to train the model. The box annotation technique followed is illustrated in Fig. 4.18.

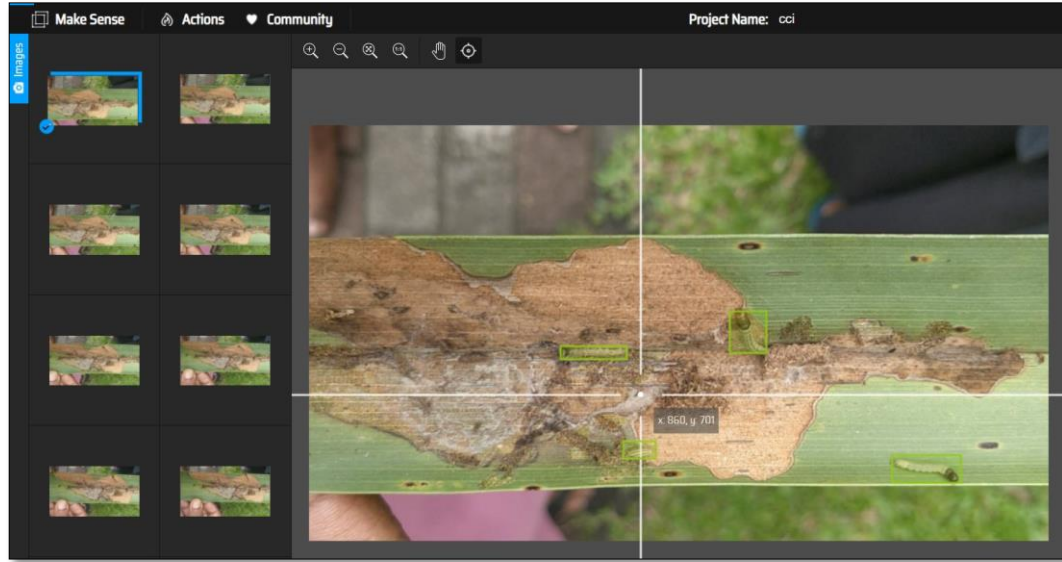


Figure 4. 18: Data annotation using makesense.ai

YOLO is a real-time object detection algorithm that uses neural networks. This algorithm is well-known for its speed and accuracy. There are several versions of YOLO, where YOLOv5 is the latest model developed by Ultralytics LLC in 2020 [47]. The YOLOv5 object detection model is also divided into four sub-versions: YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x. For this research YOLOv5x was used due to its considerably high accuracy rate compared to other versions [48].

In comparison to previous versions, the structure of YOLOv5 includes the following significant advancements that enhance the quality of object detection, especially small object detection:

1. Mosaic data augmentation
 2. Cross Stage Partial (CSP) structure
 3. Path Aggregation Network (PANet) structure
- The Mosaic data augmentation approach performs random operations on four images such as cutting, flipping, magnification, and colour range modification, and then merges the images according to the four orientations and stitches them to one image as training data. This method improves the image background and minimize the need for a large mini-batch size [49].
 - YOLOv5 has integrated CSPNet into Darknet, forming CSPDarknet as its backbone. CSPNet is used to create a more dynamic gradient composition while lowering computation time. It can effectively solve the gradient information redundancy problem in the backbone of a CNN, and incorporate gradient modifications into the feature map from start to finish, lowering the number of parameters and the FLOPS value [50]. CSPDarknet is used by YOLOv5 to extract more features from the input image.
 - PANet is used by YOLOv5 to combine features. PANet is built on Mask R-CNN and FPN, and it uses a novel FPN architecture that improves the semantics and spatial features of the feature pyramid by enhancing the path (bottom to the top) during the process of feature extraction.

Similar to Mask R-CNN, the regions of interests are marked based on IoU values (Fig. 4.7). Following non-max suppression, which ensures the object detection algorithm to detect each object only once, it returns detected objects with bounding boxes. The network architecture of YOLOv5 consists of three main parts. Backbone (CSPDarknet), Neck (PANet) and Head (YOLO layer).

First the image is entered into CSPDarknet for feature extraction, and next passed into PANet for feature fusion. Finally, the YOLO layer determines the output results of class prediction, accuracy score and localization of the object (Fig. 4.19).

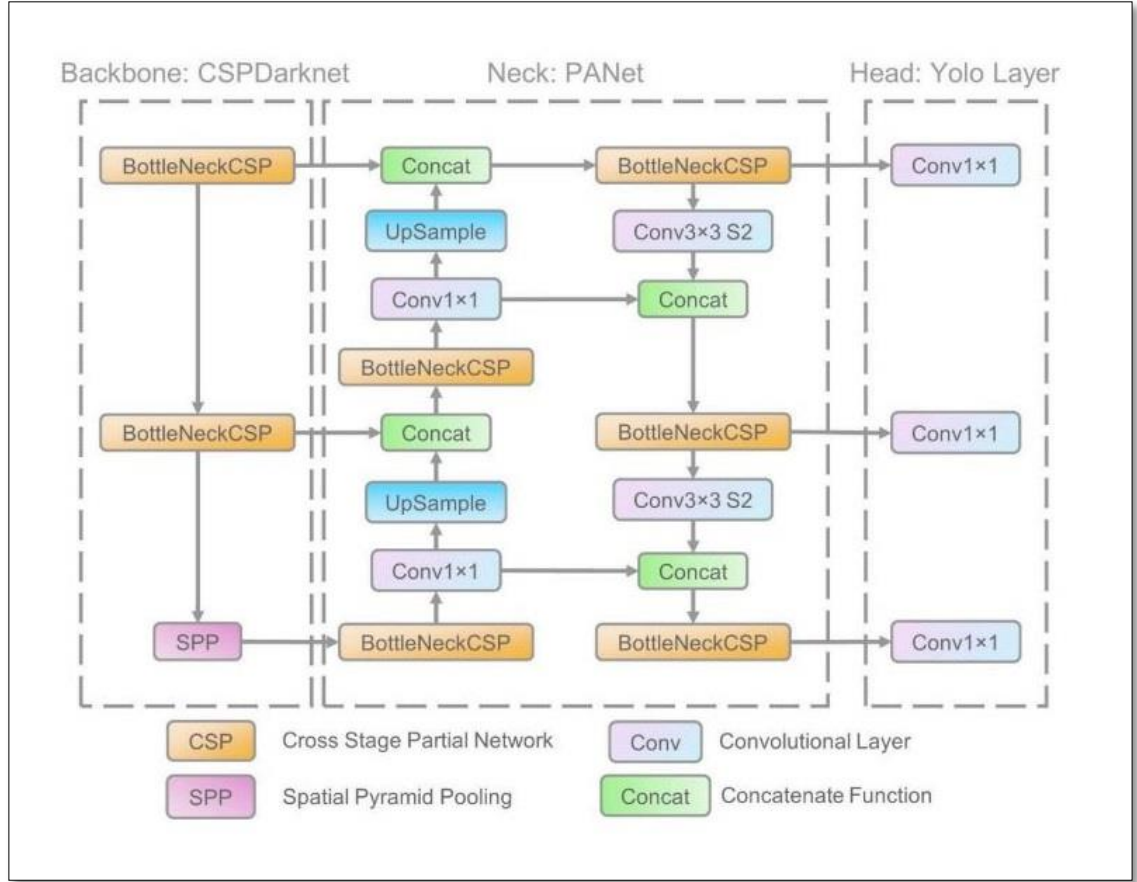


Figure 4. 19: YOLOv5 network architecture [51].

The “detect.py” file containing the detection algorithm was altered to take count of caterpillars detected for each classification and localization performed by the model. Finally, the images output by the model was stitched together using “OpenCV panorama stitching” (Fig. 4.16).

A variety of computer vision and image processing techniques were used to create the image panorama, including:

- Keypoint detection
- Match descriptors
- Homographic estimation using RANSAC
- Perspective warping

To create a panoramic image, some key points and features must be extracted (using Difference-of-Gaussian (DoG) keypoint detector and scale-invariant feature transform (SIFT) feature extractor). These features must have some unique characteristics. The overlapping points should then be used to match the relevant key points based on some measure of similarity. These overlapping points will give an indication of the orientation of the second image in reference to the first. Next, to estimate homography, RANdom SAMple Consensus (RANSAC) is used. Homography is the mapping of any two projection surfaces with the same projection center [29]. Once homography is estimated, images are warped to a common plane creating a panoramic image.

With the use of above mentioned architectures, models, algorithms and equations, the research for the identification and progression calculation of coconut caterpillar infestation was conducted.

4.1.7 Design Diagrams

System design diagrams were created to identify the essential components and organize implementations related to developing the model. Fig 4.20 and Fig 4.21 illustrates both sequence and use case diagram helped to develop this research component.

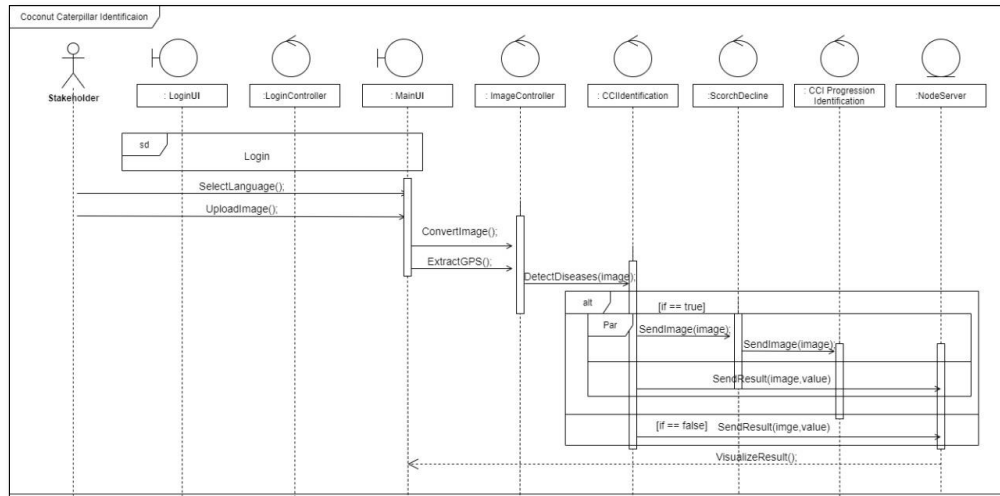


Figure 4. 20: Sequence diagram of CCI component

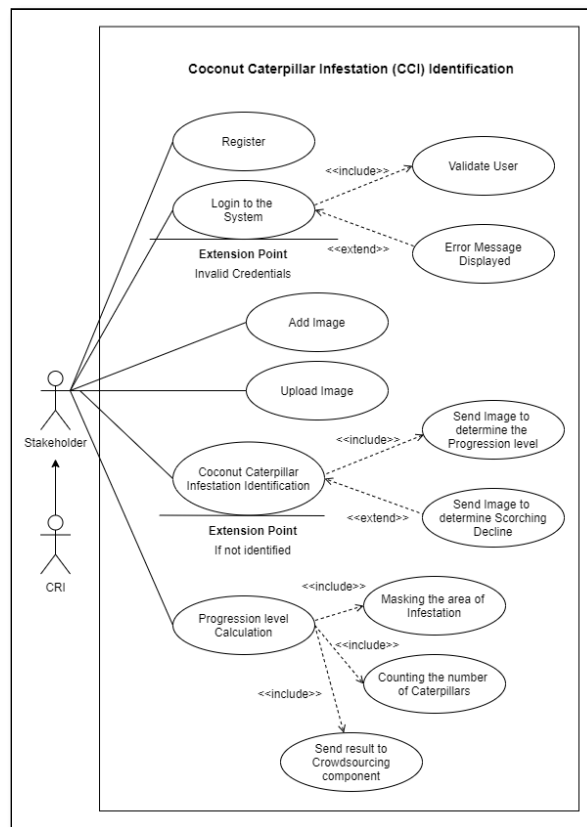


Figure 4. 21: Sequence diagram of CCI component

4.2 Commercialization aspects of the product

This research is proposed as a smart solution for one of the research gaps identified in the research programme of CRISL. The application “CocoRemedy” will be introduced to all the stakeholders such as coconut growers, estate managers of plantation companies, landowners, researchers of CRISL, CDOs of Coconut Cultivation Board (CCB), and people who are growing coconut for their daily consumption in Sri Lanka.

CRISL is hosting regular gatherings for stakeholders all around the country and the product will be pitched at those gatherings in order to commercialize it locally. Furthermore, “CocoRemedy” will be advertised on the official web page of CRISL. The application is already hosted in Google Playstore for interested users to download into their mobile devices.

The product comes in two varieties. The free version will include WCWLD and CCI classification as well as dispersion visualization. The premium version includes features such as symptom severity, progression level, real-time notifications regarding the dispersion, and the ability to contact the nearest CDO for advice on disease control measures without delay. The premium version should be purchased by stakeholders in order to determine the severity and take appropriate precautions. CRISL will support the project to continue research as well as marketing in order to globalize CocoRemedy in the future.

As future developments, detections for other prevailing pests and diseases will be added to expand the product. Furthermore, forecasting dispersions will be supported in the next release. CocoRemedy will be enhanced as an e-commerce platform where the buyers can purchase and sellers can exhibit their products. Once the product is established, a special drone system that can capture and detect the infected palms will be manufactured. CRISL and plantation companies are the main target stakeholders of this system.

5. Testing & Implementation

5.1 Implementation

CocoRemedy is a software solution comprised of both web and mobile applications. The mobile application is used to assess disease identification, classification, and progression level calculation. The results were displayed in the web application for the researchers to monitor the process. Frontend development of mobile and web applications were implemented using React Native and React Js respectively. MongoDB cluster is used as the database. Google Cloud Storage is used to store the images after detections. All the backend servers and the web application are deployed in the Google Cloud. The CCI models and algorithms were implemented using Python (version 3.6.13) on Jupyter Notebook. The implemented models were trained using Google Colab. It is a virtual machine developed with high GPU memory and RAM that is used to execute machine learning programs. In this research, the "pro" version was used for a better experience.

5.1.1 Data Preprocessing

The images collected were of various sizes (1080 x 1920). Therefore, all the images were resized to the same dimensions (416 x 416) as shown in Fig. 5.1.

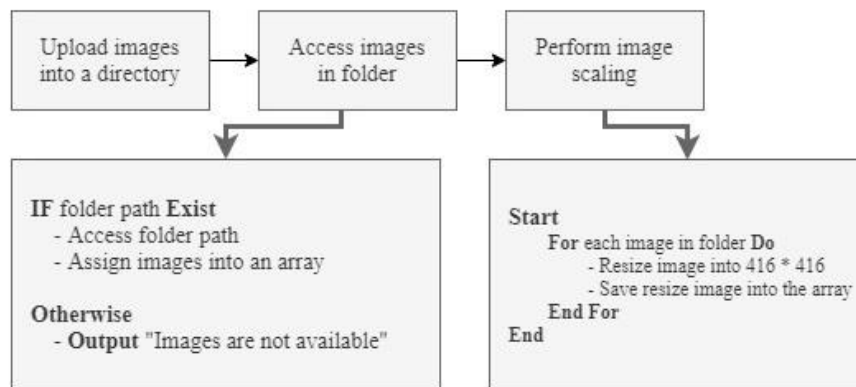


Figure 5. 1: Steps of image preprocessing

Collected images are uploaded into a local directory. The path to the local directory is accessed and checked whether images in the directory is available or not. If available, the images are assigned into an array. If not the algorithm will return a message indicating that there are no images in the folder. To open images in the directory, the Image.open() function was used. The images are saved in a variable called pillow image. The images' dimensions were changed to 416 * 416 using the resize() method. Finally, the resized images are saved using save() function. (Fig 5.2).

```
#This function will load the data from the directory
arr = os.listdir("Images/imageStitching/augment/Original_Images")

index = 1
for item in arr:

    #perform the image rescaling
    pilImg = Image.open('Images/imageStitching/augment/Original_Images/' + item ) #select the directory you need to open
    pilImg = pilImg.resize((416 ,416 ), Image.ANTIALIAS ) #set the size
    pilImg.save('Images/imageStitching/augment/Augmented_Images/ccl_' + str(index) + ".jpg") #saving path of new image
```

Figure 5. 2: Code snippet for data preprocessing (resizing)

5.1.2 Data Augmentation

Data augmentation is well recognized as an effective method for improving generalization. It converts each sample image into similar variants. After resizing, augmentation was tested using two techniques.

1. Custom augmentation algorithm
 2. Augmentation using Roboflow
- **Custom augmentation algorithm** – Initially, necessary libraries were imported (NumPy, PIL and Keras). New batch of images were created using the “ImageDataGenerator” class in Keras library and saved in a separate folders using Python Imaging Library (PIL) which supports manipulating, opening, and saving various image file formats.

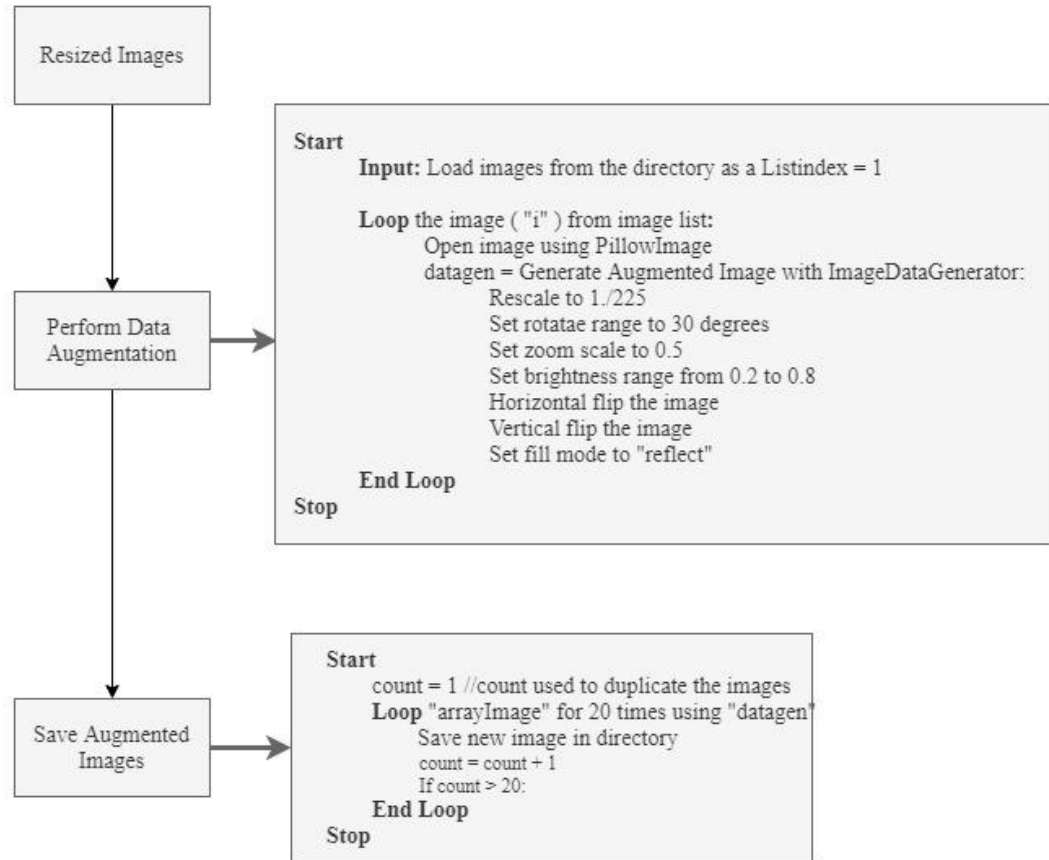


Figure 5. 3: Steps of data augmentation

The resized images were augmented into different variants using the above augmentation flow. The `ImageDataGenerator()` method accepts the original data, randomly modifies with a 30-degree rotation, shear range of 0.2, zoom range of 0.5, horizontal and vertical flip, brightness ranging from 0.2 to 0.8, and images with reflect fill mode. This process is looped for 20 times in order to generate 20 augmented image set from one image. Finally, all the images were saved in a separate folder for training. To improve code quality, preprocessing and augmentation algorithms were integrated into one (Fig 5.4 and Fig 5.5).

```

#This function will load the data from the directory
arr = os.listdir("Images/imageStitching/augment/Original_Images")

index = 1
for item in arr:

    #perform the image rescaling
    pillow = Image.open('Images/imageStitching/augment/Original_Images/' + item) #select the directory you need to open
    pillow = pillow.resize((416 ,416 ), Image.ANTIALIAS ) #set the size
    pillow.save('Images/imageStitching/augment/Augmented_Images/ccl_' + str(index) + ".jpg") #saving path of new image

    #perform the augmentation
    arrImg = img_to_array(pillow)
    arrImg = arrImg.reshape((1,) + arrImg.shape )

    datagen = ImageDataGenerator(
        rescale=1./225,
        rotation_range=30,
        shear_range=0.2,
        zoom_range=0.5,
        brightness_range=(0.2, 0.8),
        horizontal_flip=True,
        vertical_flip = True,
        fill_mode='reflect'
    )

    count = 0

    print("Augmenting image " + str(index) + " into 20 images")
    for batch in datagen.flow( arrImg, batch_size=1, save_to_dir='Images/imageStitching/augment/Augmented_Images/',
        save_prefix='ccl',
        save_format='jpeg'):

        count += 1
        if count > 20:
            break

    index = index + 1 #increment the main index

```

Figure 5. 4: Code snippet for data augmentation (python)

```

1 Start
2 Input : Load images from the directory as a List
3 index = 1
4 Loop the image ( "i" ) from image list:
5     pillowImage = Open image "i"
6     pillowImage = pillowImage.resize( 416, 416 ) //resize the image to 416x416
7     Save resized "pillowImage" in a new directory
8
9     arrayImage = convert "pillowImage" to array
10    Reshape the "arrayImage"
11
12    datagen = Generate Augmented Image with ImageDataGenerator:
13        Rescale to 1./225
14        Set rotatae range to 30 degrees
15        Set zoom scale to 0.5
16        Set brightness range from 0.2 to 0.8
17        Horizontal flip the image
18        Vertical flip the image
19        Set fill mode to "reflect"
20
21    count = 1 //count used to duplicate the images
22    Loop "arrayImage" for 20 times by generating random images with "datagen":
23        Save new image in directory
24
25        count = count + 1
26        If count > 20:
27            Exit loop
28    index = index + 1
29 Stop

```

Figure 5. 5: Code snippet for data augmentation (pseudocode)

- **Roboflow** [52] - Each image was augmented into three variants using flip (horizontal and vertical), rotation, and shear (horizontal and vertical). (Fig. 5.6).

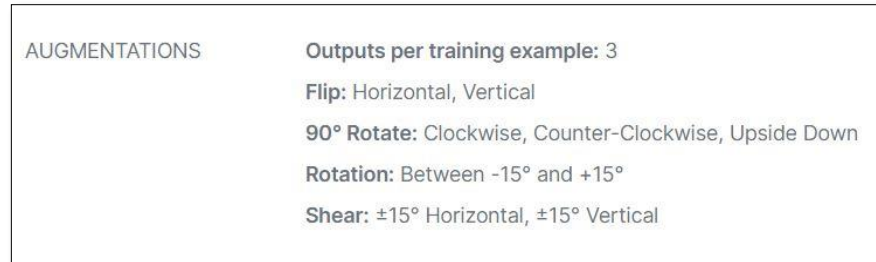


Figure 5. 6: Augmentation techniques applied using Roboflow

5.1.3 Deep learning model implementation

- **Infestation identification and classification (Mask R-CNN model)**

Fig 5.7 illustrates an overall training process of Mask R-CNN model.

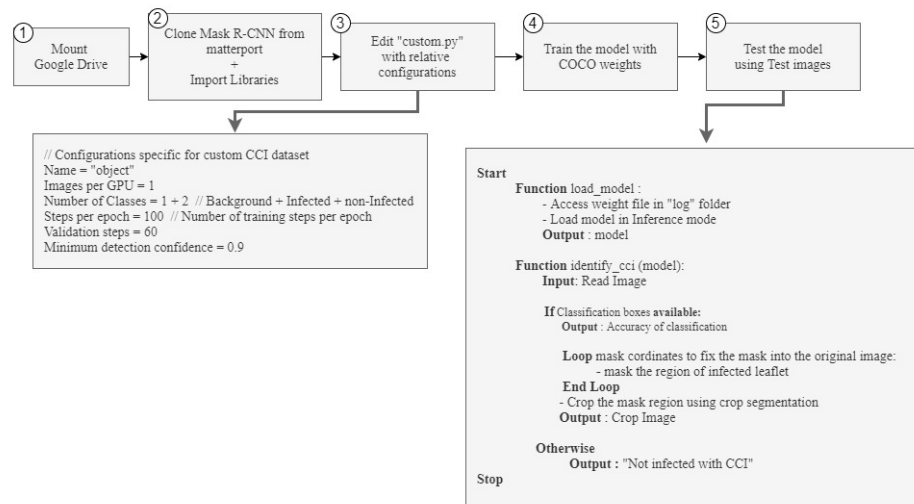


Figure 5. 7: Steps of CCI identification and classification

The Notebook file containing the python algorithms were uploaded to Google Colab for training. Since the training requires the use of a graphics card, the GPU is enabled. Initially, the Mask R-CNN repository with all the configuration files were cloned^①(Fig. 5.8) from GitHub (matterport) along with relevant libraries (Tensorflow & Keras) (Fig. 5.9).

```
# Mount Google Drive and Clone Mask R-CNN repository from matterport
from google.colab import drive
drive.mount('/content/drive')
%cd '/content/drive/MyDrive/Colab Notebooks/maskrcnn'
!git clone --quiet https://github.com/matterport/Mask_RCNN.git
```

Figure 5. 8: Cloning Github repository

```
# Install required Libraries
%cd '/content/drive/MyDrive/Colab Notebooks/maskrcnn/Mask_RCNN'
!pip install -q PyDrive
!pip install -r '/content/drive/MyDrive/Colab Notebooks/maskrcnn/Mask_RCNN/requirements.txt'
!pip3 uninstall keras-nightly
!pip3 uninstall -y tensorflow
!pip3 install keras==2.1.6
!pip3 install tensorflow==1.15.0
!pip3 install h5py==2.10.0
```

Figure 5. 9: Code snippet for installing libraries needed for training

The “custom.py” file which contains the model configurations were edited^③accordingly (Fig. 5.10) to best fit CCI classification through series of hyper-parameter tuning. The configurations were edited with 2 images per GPU, three classes (with CCI, without CCI, and background), 100 steps per epoch, 100 validation steps and a minimum detection confidence of 0.9. The input image sizes were 416 x 416.

```

class CustomConfig(Config):
    """Configuration for training on the CCI dataset.
    Derives from the base Config class and overrides some values.
    """

    # Give the configuration a recognizable name
    NAME = "object"

    # We use a GPU with 12GB memory, which can fit two images.
    # Adjust down if you use a smaller GPU.
    IMAGES_PER_GPU = 2

    # Number of classes (including background)
    NUM_CLASSES = 1 + 2 # Background + CCI + non-CCI

    # Number of training steps per epoch
    STEPS_PER_EPOCH = 100

    # Skip detections with < 90% confidence
    DETECTION_MIN_CONFIDENCE = 0.9

```

Figure 5. 10: Editing training configurations in “custom.py” file

Transfer learning was used to train the COCO weights in order to ensure high accuracy of the results obtained when the actual data set is used. The training code④snippet for the Mask R-CNN model is shown in Figure 5.11.

```

# Initiate training process
!python3 custom.py train --dataset=/content/drive/MyDrive/Colab\ Notebooks/maskrcnn/Dataset --weights=coco

```

Figure 5. 11: Code snippet to start the training process

After a few seconds, we may see if a first template (h5 file) is ready. The h5 files will be downloaded inside the log folder as shown in Fig. 5.12

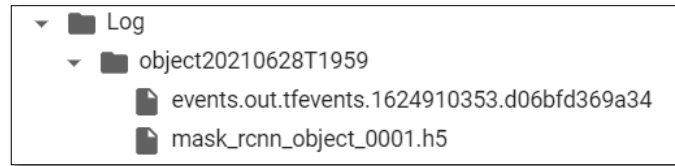


Figure 5. 12: Created weight files

After training the model, the weights are loaded from the “log” folder (using load_model function) to detect images^⑤

```
def load_model():
    #LOAD MODEL. Create model in inference mode
    model = mdelib.MaskRCNN(mode="inference", model_dir=MODEL_DIR, config=config)

    # Load COCO weights Or, Load the Last model you trained
    weights_path = WEIGHTS_PATH
    # Load weights
    print("Loading weights ", weights_path)
    model.load_weights(weights_path, by_name=True)

    return model
```

Figure 5. 13: Code snippet for loading the model

The best fitted weights created from the model is provided. Using the weights, classification and localization on test images determine the confidence score of the detection. The algorithm checks whether there are any labels (classification boxes) with “CCI”. If the image was identified as infected it is further sent to calculate the progression level. If not the system will output that the leaf entered was not infested with CCI.

Using the mask coordinates and dimensions provided by the FCN layer of Mask R-CNN, the leaf is only extracted while the background is separated (crop segmentation). Finally, the crop segmented image is used to calculate the percentage of damaged area with compared to the healthy regions. This is a clear example which demonstrates the effectiveness of instance segmentation in Mask R-CNN.

To apply the minimal number of clusters, the colour ranges in the original images were reduced to three. The range of green colours were turned into one RGB value. Similarly, the brown colour caused due to coconut caterpillar damage was also converted to single RGB pixel value. The rest of the colours were converted to white (background) before applying k-means clustering algorithm. Fig. 5.13 depicts the overall workflow of progression level calculation.

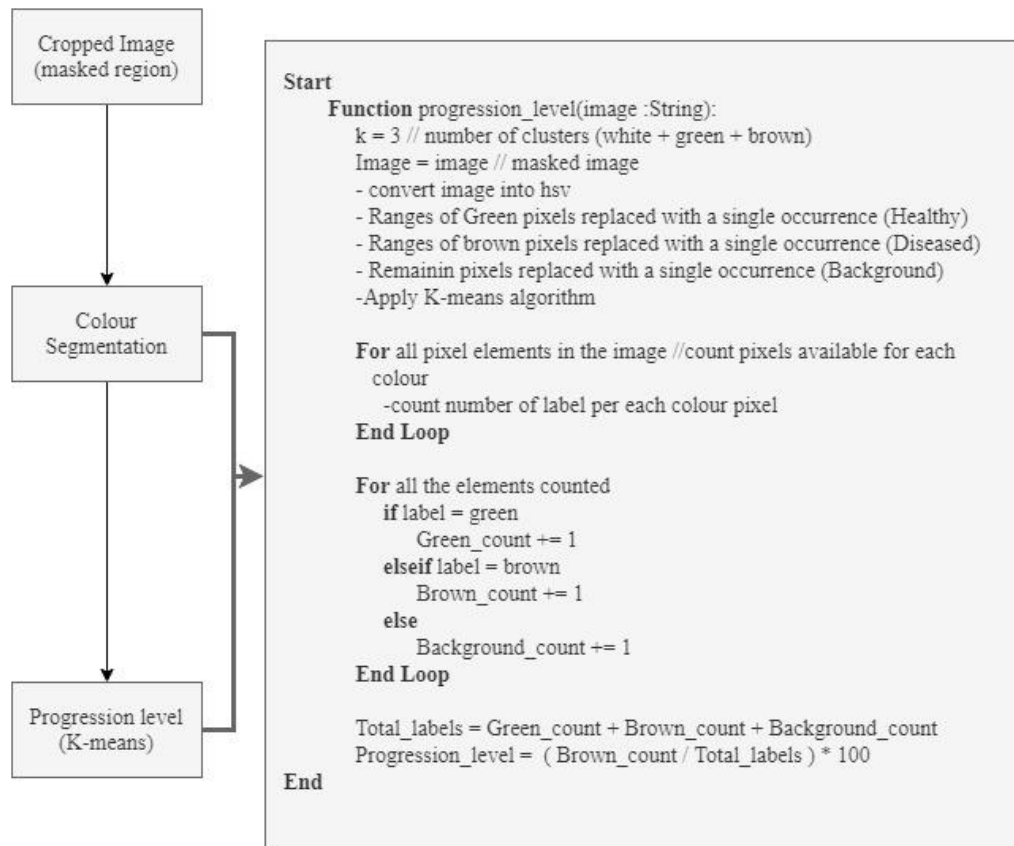


Figure 5. 14: Overview of progression level calculation

- **Identification and counting caterpillars on coconut leaflets (YOLOv5)**

The overall training process of YOLOv5 is somewhat similar to that of Mask R-CNN. Both are object detection, classification and localization models used in deep learning.

The major difference is the availability of a FCN layer in Mask R-CNN to instance segment images. However, as the name suggests YOLO algorithms are comparatively fast object detection algorithms [53]. Furthermore, YOLOv5 has a less inference time as it uses the PyTorch Architecture. The overall workflow diagram used to train the YOLOv5 object detection model is illustrated in Fig 5.14.

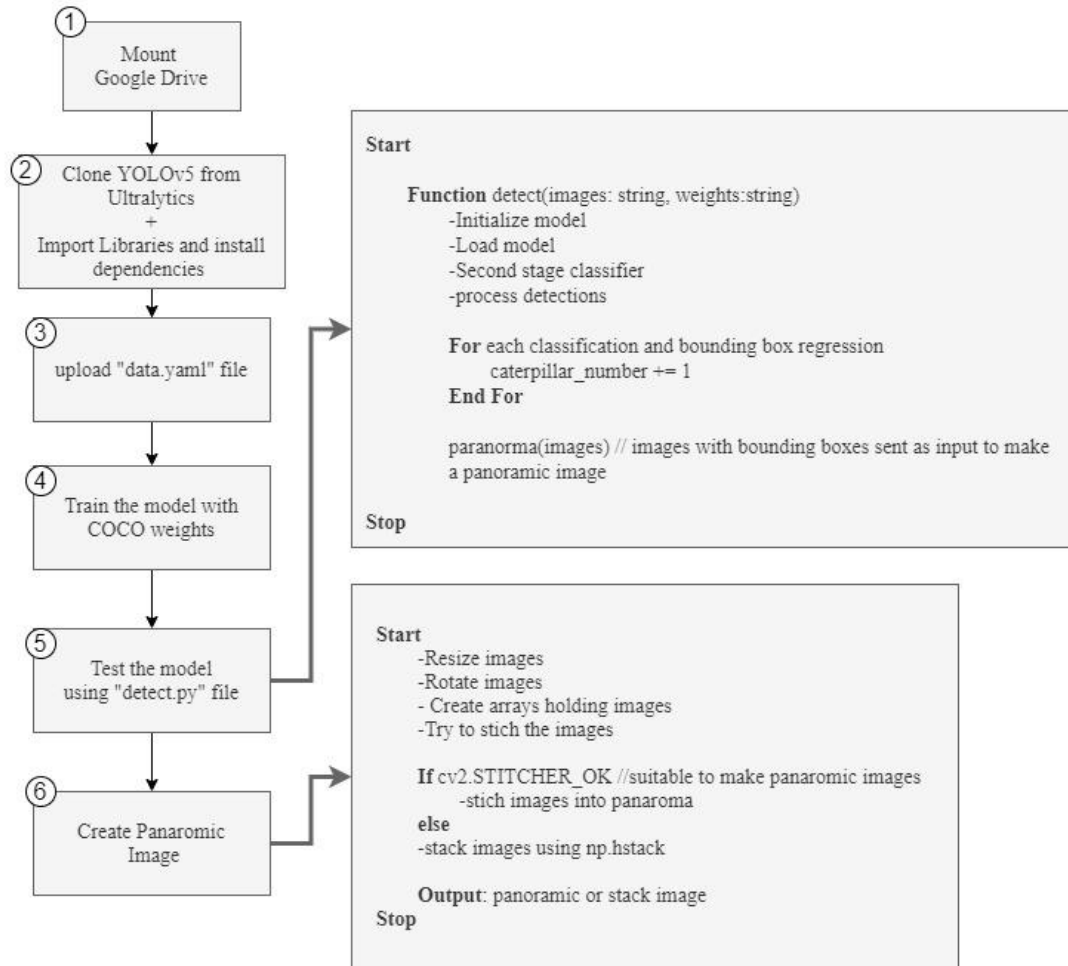
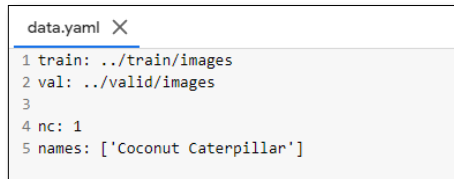


Figure 5. 15: Overview of caterpillar calculation using YOLOv5

After cloning YOLOv5 from Ultralytics, a “yaml” file should be uploaded which contains the training, validation image paths as well as number of classes with class labels 3. Custom yolov5x.yaml file should also be modified with number of classes prior training the model.



```
data.yaml X
1 train: ../train/images
2 val: ../valid/images
3
4 nc: 1
5 names: ['Coconut Caterpillar']
```

Figure 5. 16: data.yaml file

For testing, the developed model use the detect() algorithm which is inside detect.py python file. The predefined algorithm does not contain a method to count each prediction boxes made. Therefore, to calculate the caterpillars, a custom python code was written inside the detect() algorithm as shown in Fig 5.14. 5

Finally, the images detected with bonding boxes are passed to the panorama() function to create panoramic images.



```
def panorama(first,second):
    heightImg = 1024
    widthImg = 768

    #Resize the images from your directory
    first = cv2.resize(first, (widthImg, heightImg)) # Resize the first image
    second = cv2.resize(second, (widthImg, heightImg)) # Resize the first image

    #Rotate the image
    firstR = cv2.rotate(first, cv2.ROTATE_90_COUNTERCLOCKWISE)
    secondR = cv2.rotate(second, cv2.ROTATE_90_COUNTERCLOCKWISE)

    panoramaR = []
    panoramaR.append(firstR)
    panoramaR.append(secondR)

    panorama = []
    panorama.append(first)
    panorama.append(second)

    #Stitch the image into a panorama
    stitcher = cv2.Stitcher.create()
    ret, pano = stitcher.stitch(panoramaR)

    if ret==cv2.STITCHER_OK:
        cv2.imwrite('Pano.jpg', pano)
        cv2.waitKey()
        cv2.destroyAllWindows()
    else:
        stitcher = cv2.Stitcher.create()
        ret, pano = stitcher.stitch(panorama)

        if ret==cv2.STITCHER_OK:
            cv2.waitKey()
            cv2.destroyAllWindows()
            print('panorama created')
        else:
            stackImage = np.hstack((firstR, secondR))
            print(pano)
            cv2.imwrite('Stack.jpg', stackImage)
            cv2.waitKey()
            cv2.destroyAllWindows()
```

Figure 5. 17: Code snippet used to create panoramic images

This panoramic image is displayed to users in the mobile application. It arranges the images and try to stich them together using keypoints. If the images are possible to stitch together (STITCHER_OK) panoramic image is created. If not the images are stacked together without keypoint detection using hstack() method.(6)

- **Counting caterpillars on white papers (image processing)**

The manual process was also automated using image processing techniques, providing users to choose their preferred method of calculating caterpillars. The overall process is illustrated in Fig 5.17

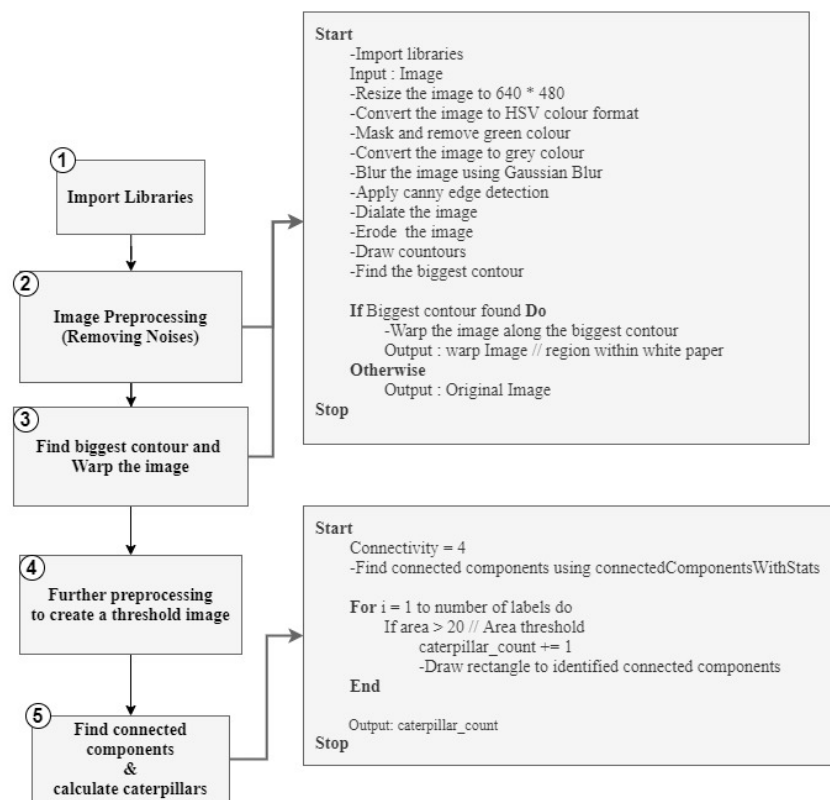


Figure 5. 18: Overview of caterpillar calculation image processing

First, libraries were imported. The image was resized in the dimensions of 640 x 480 and converted into HSV colour format. Gaussian blur was applied to the greyscale image with a kernel size of 5 for both height and width. After blurring, canny edge detection was used with 200 as high threshold value and 255 as low threshold value of intensity gradient. The image is dilated with 2 iterations and eroded with 1 iteration. Next, the biggest contour of the image was analyzed to warped excluding the background except the region which belongs under the white paper.

```
# FIND ALL CONTOURS
imgContours = img.copy() # Copy image for display purposes
imgBigContour = img.copy() # Copy image for display purposes
contours, hierarchy = cv2.findContours(imgErode, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE) # Find all contours
cv2.drawContours(imgContours, contours, -1, (0, 255, 0), 10) # Draw all detected contours

# FIND THE BIGGEST CONTOUR AND WARP THE IMAGE
biggest, maxArea = utils.biggestContour(contours) # Find the biggest contour

if biggest.size != 0:
    biggest = utils.reorder(biggest)
    cv2.drawContours(imgBigContour, biggest, -1, (0, 255, 0), 20) # Draw the biggest contour
    imgBigContour = utils.drawRectangle(imgBigContour, biggest, 2)
    pts1 = np.float32([biggest]) # Prepare point for wrap
    pts2 = np.float32([[0, 0], [widthImg, 0], [0, heightImg], [widthImg, heightImg]]) # Prepare point for wrap
    matrix = cv2.getPerspectiveTransform(pts1, pts2)
    imgWarpColored = cv2.warpPerspective(img, matrix, (widthImg, heightImg))

    # REMOVE 20 PIXELS FROM EACH SIDE
    imgWarpColored = imgWarpColored[20:imgWarpColored.shape[0] - 20, 20:imgWarpColored.shape[1] - 20]
    imgWarpColored = cv2.resize(imgWarpColored, (widthImg, heightImg))
    cv2.imwrite('sanned.jpg', imgWarpColored)
else:
    plt.imshow(imgResizeCopy)
```

Figure 5. 19: Code snippet to find the biggest contour and warp the image

After extracting the white paper, it is easier to count the caterpillars (since background noises are removed). For further preprocessing, the image was threshold using Adaptive thresholding. Finally, connected components of the images were identified. For each connected component a counter was maintained to calculate the total number of caterpillars (Fig 5.19).

```
connectivity = 4
# Perform the operation
output = cv2.connectedComponentsWithStats(eroded_image, connectivity, cv2.CV_32S)
caterpillar_count = 0

for i in range(1, num_labels):
    if stats[i][4] > 20: # Area threshold
        caterpillar_count += 1
        left_x = stats[i][0]
        up_y = stats[i][1]
        right_x = left_x + stats[i][2]
        down_y = up_y + stats[i][3]
        cv2.rectangle(imgWarpColored, (left_x, up_y), (right_x, down_y), (0, 255, 0), 3)
```

Figure 5. 20: Code snippet used to find the connected components and calculate caterpillars

5.2 Testing

The final step in the process is to test the developed models. During the testing phase, the performance of the developed models is evaluated using images from the test dataset. 160 and 140 test images were used to test the classification of CCI and Caterpillar calculation models respectively. Several images of caterpillars on white paper were used to test the image processing algorithms. Furthermore, the integrated system was tested in local and cloud environments. Several tests were performed until all defects were identified and resolved, and the system was ready to be deployed in the production environment. AWS Ec2 instance and Google cloud virtual machine were used to test the integrated system in the production environment, while Google Colab (pro version) and local machine were used to test models locally.

5.2.1 Test Plan and Test Strategy

Test planning is a blueprint created for carrying out the tests required to ensure that the software is effective. It is a baseline plan created that includes a list of tasks, scope and objectives for tracking the project's progress. A test strategy is a series of steps and procedures that regulates the software testing process. It outlines the elements and functions to be tested in relation to the risks they provide to users.

Steps and procedures in test strategy:

- Define the items to be tested
- Select the functions based on the importance and risk on user
- Design test cases as identified by the use case description
- Execute
- Record results
- Identify bugs
- Correct bugs
- Repeat the test case until expected results are met

5.2.2 Test Case Design

The following test cases were designed to ensure system reliability by testing all system functionalities.

Table 5. 1: Test case to verify whether the captured image is stored Google cloud storage.

Test Case Id	01
Test Case	Verify image upload
Test Scenario	Verify whether the captured image is stored Google cloud storage (Firebase bucket)
Input	Captured suspicious leaflet images
Expected Output	1. 200 status code should be displayed 2. The images must be stored in the firebase bucket
Actual Result	1. 200 status code was displayed 2. The images were stored in the firebase bucket
Status (Pass/Fail)	Pass

Table 5. 2: Test case to classify and select the best model for CCI (Mask R-CNN)

Test Case Id	02
Test Case	Classification of CCI using Mask R-CNN
Test Scenario	Testing images to classify CCI and select the best model.
Precondition	2560 labelled training & 640 validation images
Input	Test images (with and without CCI)
Expected Output	High accuracy (mAP value)
Actual Result	High model accuracy with 95.31% mAP value
Status (Pass/Fail)	Pass

Table 5. 3: Test case to mask the infested leaflets

Test Case Id	03
Test Case	Application of instance segmentation in Mask R-CNN
Test Scenario	Masking the area of infested leaflets
Precondition	Trained model (Mask R-CNN) with high predication accuracy
Input	CCI infested leaflet
Expected Output	Accurately masked leaflet
Actual Result	The area of the entire leaflet with CCI is masked
Status (Pass/Fail)	Pass

Table 5. 4: Test case to crop segment the masked area

Test Case Id	04
Test Case	Crop Segmenting masked region
Test Scenario	Extract only the mask region by excluding the background
Precondition	Masked image with mask coordination
Input	CCI infested leaflet (masked)
Expected Output	Image with only the masked area
Actual Result	A new image was created with only the masked area while the background was excluded (blackened)
Status (Pass/Fail)	Pass

Table 5. 5: Test case to calculate the level of progression

Test Case Id	05
Test Case	Progression level calculation
Test Scenario	Calculating the ratio between green and brown pigments caused by the damage
Precondition	Colour segmented image containing only the CCI leaflet without the background
Input	CCI infested leaflet (colour segmented)
Expected Output	The percentage of brown area compared with the leaflet
Actual Result	Percentage of progression throughout the leaflet was calculated accurately without neglecting even the small patches
Status (Pass/Fail)	Pass

Table 5. 6: Test case to detect true negative results

Test Case Id	06
Test Case	Testing leaflets without CCI
Test Scenario	Testing accurate classification (true negative) of non-infested leaflets
Precondition	Trained model (Mask R-CNN) with high predication accuracy
Input	Leaflet without CCI (healthy or other diseased)
Expected Output	Correctly classify as not infested
Actual Result	The leaflet was identified and labeled as non-infested
Status (Pass/Fail)	Pass

Table 5. 7: Test case to calculate caterpillars on leaflets

Test Case Id	07
Test Case	Caterpillar classification and calculation on leaflets
Test Scenario	Caterpillars are identified and calculated on their natural habitat (coconut leaflets)
Precondition	Trained model (YOLOv5) with high predication accuracy
Input	Leaflet with and without caterpillars
Expected Output	Correctly calculate the number of caterpillars
Actual Result	All the caterpillars are calculated even the ones who are very hard to identify (hidden inside galleries). If there are no caterpillars the count was given as non
Status (Pass/Fail)	Pass

Table 5. 8: Test case to calculate caterpillars on white paper

Test Case Id	08
Test Case	Caterpillar calculation on white paper (manual method)
Test Scenario	Caterpillars are marked and calculated on white papers
Precondition	Extracted caterpillars into the paper
Input	Paper with and without caterpillars
Expected Output	Correctly calculate the number of caterpillars
Actual Result	All the caterpillars are calculated by removing the dust and soil particles that come along with it.
Status (Pass/Fail)	Pass

Table 5. 9: Test case to create panoramic image

Test Case Id	09
Test Case	Stitching images into a panoramic image
Test Scenario	Caterpillar calculated images on leaflets are stitched together to create a panoramic image
Precondition	Capture two areas of the infested leaflet
Input	Two images of caterpillar counted areas of a leaflet
Expected Output	Panoramic image
Actual Result	The images are stitched together accurately by connecting the keypoints of provided images
Status (Pass/Fail)	Pass

Table 5. 10: Test case to check the integrated system

Test Case Id	10
Test Case	Testing the integrated mobile application
Test Scenario	Testing identification, classification, progression level calculation and caterpillar calculation from the mobile application
Precondition	System should be integrated connecting all the research components
Input	Leaflet image captured in the field
Expected Output	Correct identification, classification, progression level determination and number of caterpillars
Actual Result	All the results were accurately displayed without any error

Status (Pass/Fail)	Pass
--------------------	------

6. RESULTS AND DISCUSSIONS

6.1 Results

6.1.1 Infestation identification and classification

The identification, classification and progression level determination of CCI was achieved using Mask R-CNN and K-means clustering algorithm. The trained model was able to accurately distinguish both infested and non-infested leaflets (healthy and other diseases) Fig 6.1 illustrates the results of instance segmentation (masking) of both infested and non-infested leaflets.



Figure 6. 1: Mask R-CNN instance segmentation and classification of leaflets

Using the Faster R-CNN layer, identification, classification and bounding-box regression was performed to mark the region of leaflets with labels. A rectangular box is drawn around the object after classification. Finally, the FCN layer masks the images according to the coordinates (Fig 6.1).

6.1.2 Identification of the Degree of Diseased Condition in Leaves

After the leaf is segmented (a), the class label of the image is checked. If the label identifies the image as infected, crop segmentation is performed to separate the mask region (leaflet) from the background (b). Using the mask coordination and dimensions the leaf is only extracted while the background is separated (Fig 6.2). This is a clear example where instance segmentation of Mask R-CNN is useful.

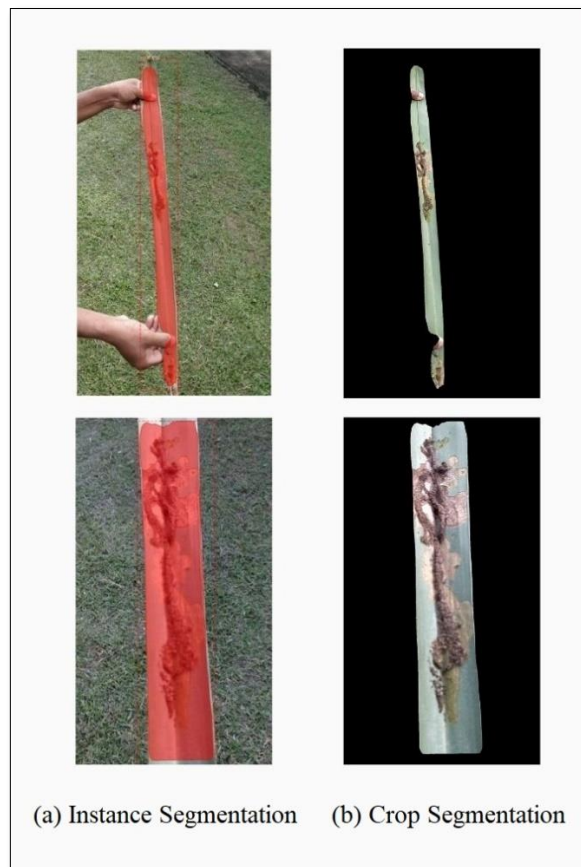


Figure 6. 2: Results of crop segmentation

After crop segmenting, the HSV range of green colours were turned into one RGB value. (10. 255. 20) Similarly, the brown colour caused due to the damage of caterpillars (necrotic regions) were also converted to single white RGB colour (139. 69. 19.). The rest of the colours were turned to white before applying k-means clustering algorithm. The reason for the conversion is to minimize the number of clusters needed (Fig 6.3).

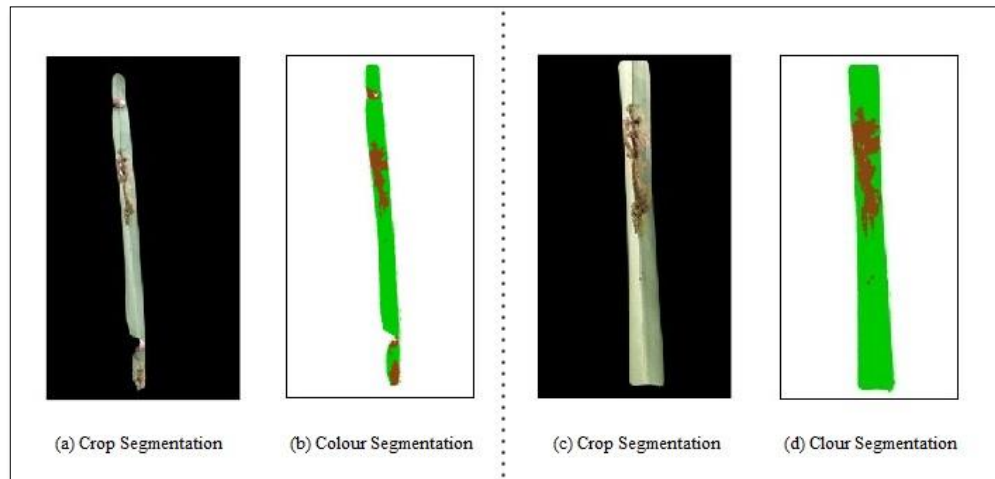


Figure 6. 3: Results of colour segmentation

Finally, using the 3 clusters of colours (white, green and brown), the amount of brown pixels with respect to the leaf area (brown + green) is calculated. Since all pixels are analyzed, even the small patches are calculated making the solution more reliable. Fig 6.4 illustrate the final outcome of the calculation for (b) in Fig 6.3.

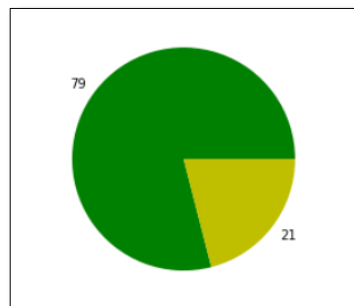


Figure 6. 4: Results of progression level calculation

A summarized flow of infestation identification classification and progression level calculation is shown in Fig 6.5.

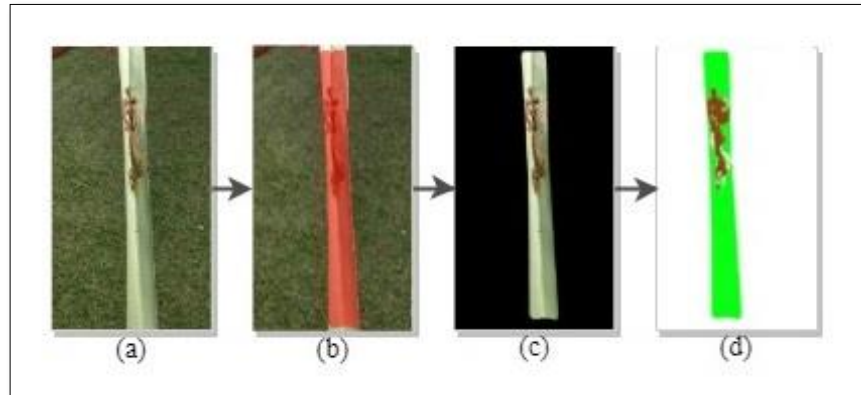


Figure 6. 5: Summarized result of CCI classification and progression level calculation

Suspicious leaflets (a) were sent through the custom Mask R-CNN model for detection. Using the model leaflets were masked (b) and extracted from the background using crop segmentation (c). Finally, colour segmentation (d) was applied to find the extent of damage caused by coconut caterpillars. Range of HSV colours that belongs to both green and brown (necrotic leaf area due to feeding of caterpillars) was replaced with a single occurrence (pixel values) before using K-means clustering algorithm to calculate the ratio between healthy and damaged parts of the leaflet (Fig. 6.5).

6.1.3 Identification and counting caterpillars (YOLOv5)

Caterpillars residing in the leaflets were annotated in order to train the object detection model. Following training, YOLOv5 detection algorithm generated output for both categorization and localization of caterpillars in leaflets. Even caterpillars that were not clearly visible were correctly recognized. Finally, the number of caterpillars was counted for each prediction made by changing the detecting algorithm. Figure 6.6 shows a sample image of identified caterpillars as well as the result of calculating number of caterpillars available in the image.

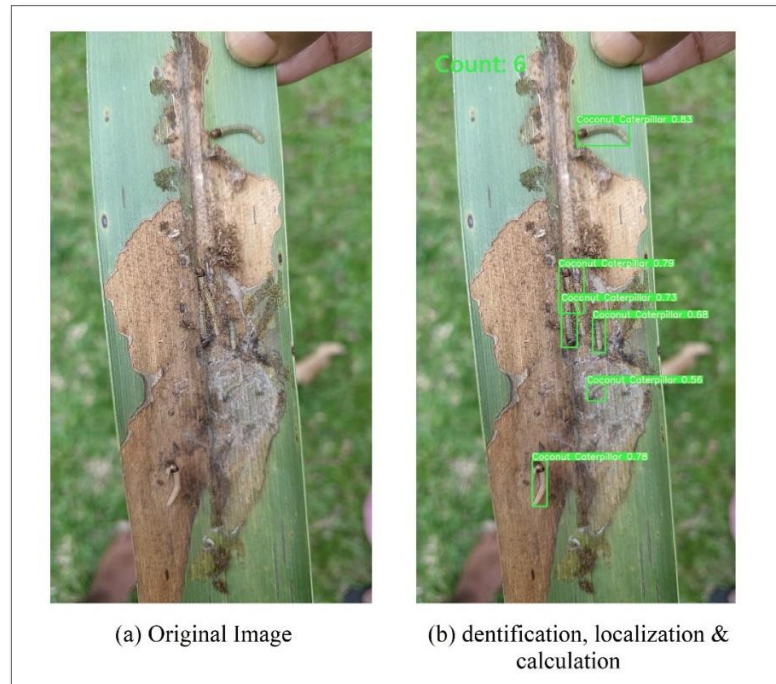


Figure 6.6: Results of caterpillar calculation using YOLOv5

The system accepts two images of infected areas for calculation since coconut leaf is long. Those two images are stitched together into a panoramic image considering the keypoints of images (Fig 6.7).

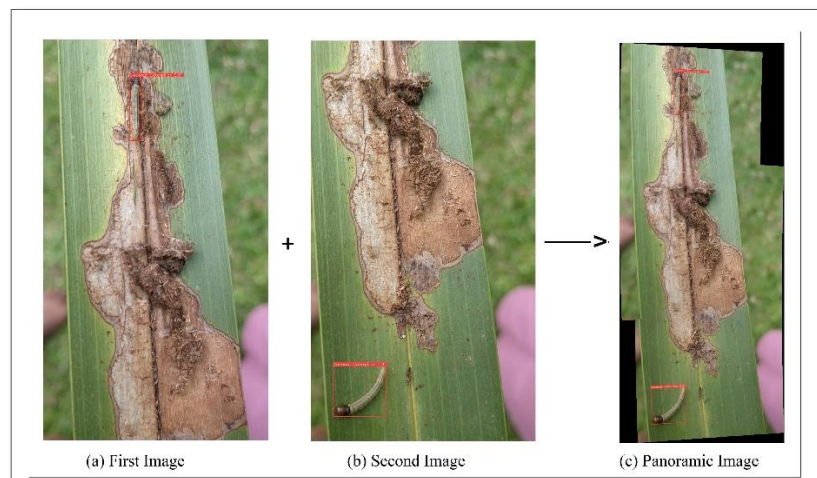


Figure 6.7: Results of creating panoramic image

If the keypoints are not detected the images are stacked together (Fig 6.8).

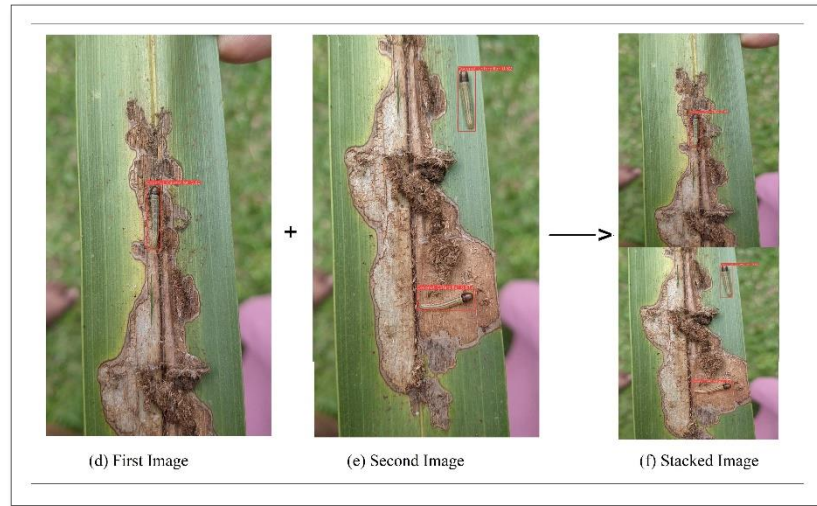


Figure 6. 8: Results of creating stacked image

6.1.4 Identification and counting caterpillars (Image Processing)

Although calculating caterpillars using YOLOv5 is efficient, some agricultural practitioners still prefer the manual method (calculating on a white paper). To support their needs, the manual calculation method was also automated using image processing techniques

Initially the white paper is scanned to exclude the surroundings. The captured image is resized (a) and background colours (around the paper) were removed (b). Using Canny edge detection largest contour (paper margin) was found. Next, using the largest contour the image is warped to exclude the background and keep the area under the white paper (d). After a series of image processing techniques, small particles (dust and soil) were removed using erosion (g). Finally, by finding the connected components, number of caterpillars were calculated (h). Fig. 6.9 shows the overall results of this calculation process.

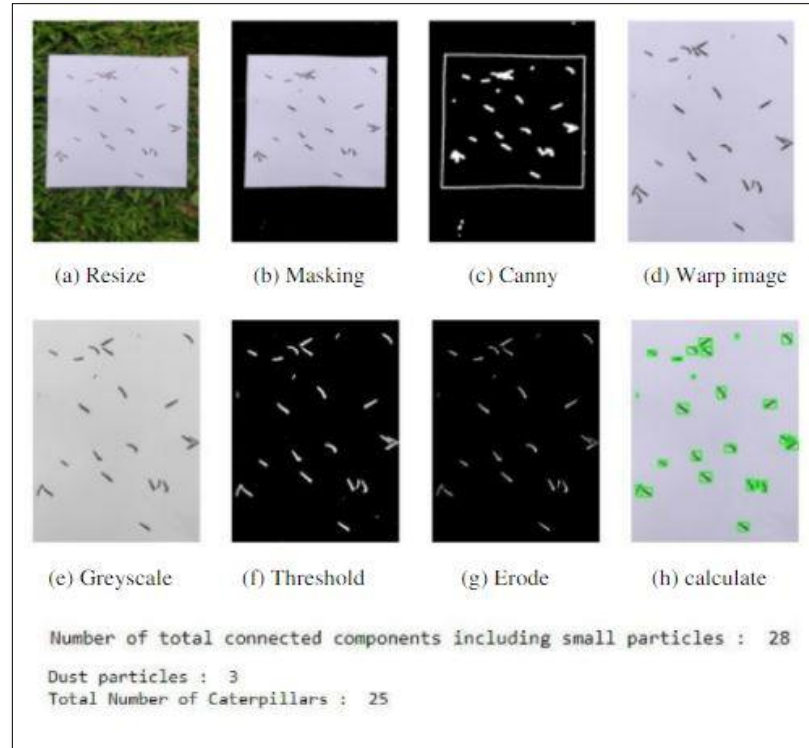


Figure 6. 9: Results of calculating caterpillars using image processing

6.1.4 Results of evaluating deep learning models

The basic quantitative assessment for object detection is the calculation of IoU values. It evaluates the similarities between the predicted anchor boxes and the ground truth bounding box (Fig. 4.7). The object is considered only if the IoU value is greater than 0.5.

For the evaluation of models, performance was assessed by comparing the annotated images with the prediction results during the training process by considering the loss values. After training metrics such as precision, recall, F1 score, and Average Precision (AP) was used assess the performance. Precision (5) measures the number of true predictions made by the model for a single class image [31]. Recall is the model's ability to predict positive data.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (5)$$

Where, TP is the number of positive samples identified while, FP is the number of false positive samples. The precision and recall values calculated for both Mask R-CNN and YOLOv5 object detection model is shown in Fig 6.10. Both the models have performed well in identifying true predictions. Precision rate of 92% was achieved by Mask R-CNN while 95% by YOLOv5 model

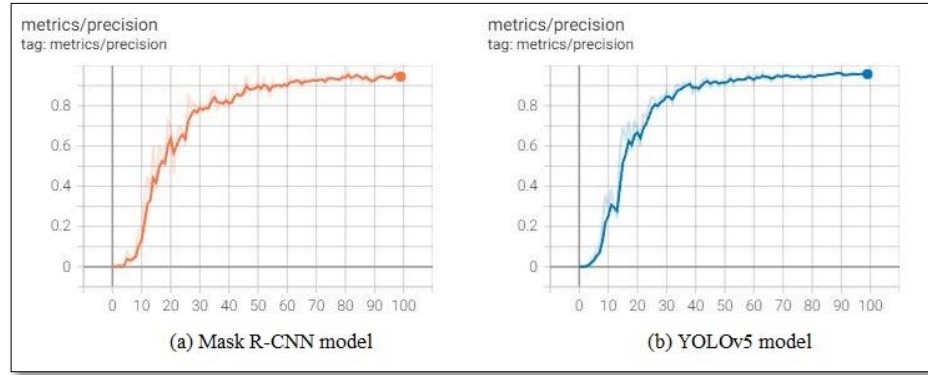


Figure 6. 10: Precision graphs for Mask R-CNN and YOLOv5 models

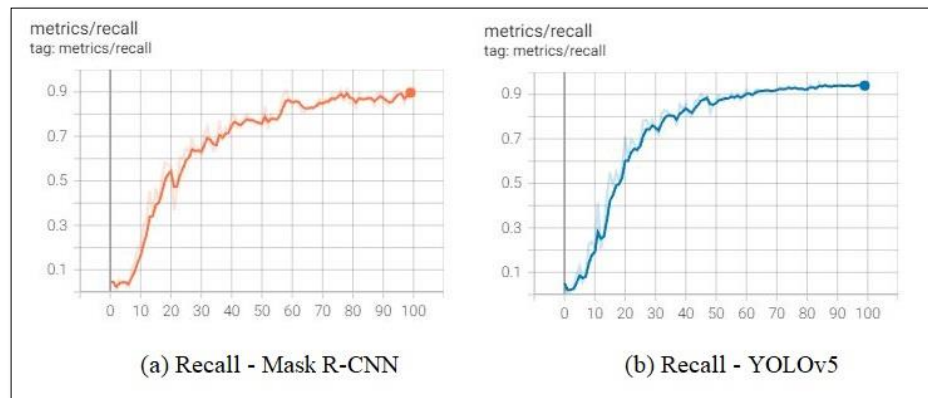


Figure 6. 11: Recall graphs for Mask R-CNN and YOLOv5 models

Average precision (6) for all images in a particular class is defined as

$$\text{Average Precision} = \frac{\sum \text{Precision}}{N'} \quad (6)$$

Where, \sum Precision is the sum of precision for all the images and N' is the number of images. Finally, with the use of above metrics Mean Average Precision (mAP) is calculated. mAP is a popular evaluation metric for object detection (localisation and classification), which is often used to assess the performance of deep learning models. If the dataset contains multiple classes, a single number is required to assess the performance of the model. As a result, the mAP value is calculated. The mean average of the calculated average precision is denoted by mAP (7). \sum Average Precision is the sum of average precision and N' is the number classes

$$\text{mAP} = \frac{\sum \text{Average Precision}}{N'} \quad (7)$$

Fig 6.12 illustrates the accuracy graphs (mAP) for both Mask R-CNN and YOLOv5 models.

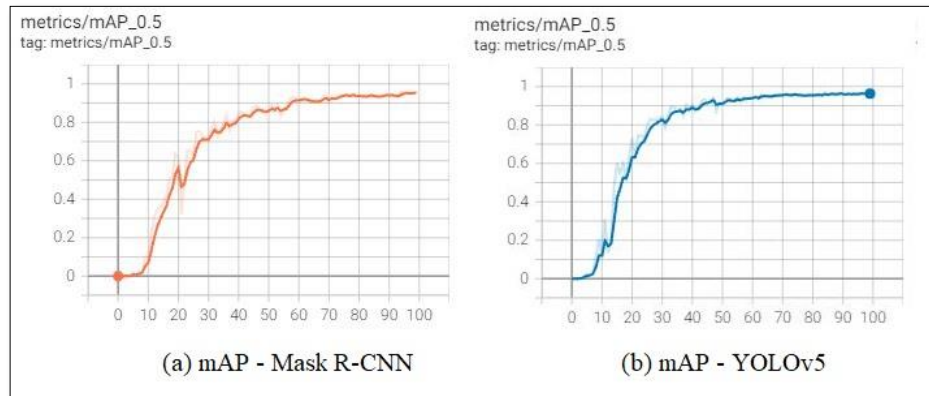


Figure 6. 12: mAP graphs for Mask R-CNN and YOLOv5 models

In the Mask R-CNN model, ResNet-101 achieved high detection rates with a mAP value of 95.26%. The trained Darknet-based YOLOv5 model achieved a mAP value of 96.28%.

The F1-score is calculated by dividing the product of recall and precision by the total value of recall and precision.

$$F1 - Score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \quad (8)$$

6.1.5 Product deployment

The finalized web application was deployed in Google Cloud, while the mobile application was deployed on Google Play Store.

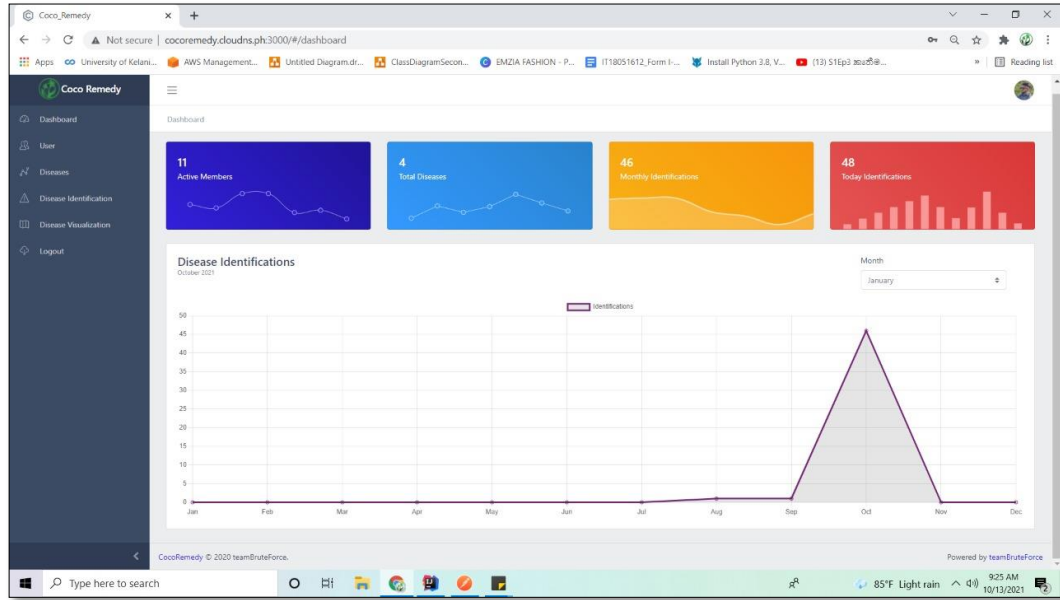


Figure 6. 13: Deployed web application



Figure 6. 14: Deployed mobile application

6.2 Research Findings

Results of the preliminary survey revealed that about 70% of general public are unaware of the coconut caterpillar as a pest, that can cause devastating damage to coconut palm, leading to substantial economic losses. A smart solution/mobile application was developed which can be used to educate growers while facilitating identification of coconut caterpillar damage and the severity of infestation at the field. On the same time the researches at CRI is provided with a data base on locations, climatic data of the location where the infestation is identified. The application also provides platform for communication between growers and experts on the control measures of the pest. The main users of the product are coconut growers, estate managers and field officers of plantation companies, landowners, CDOs of CCB, CRISL, and also people who grow coconut for their daily consumption in Sri Lanka.

Input data of the application is identified as an image taken from lower side of an infested leaflet. Data augmentation and preprocessing techniques were leads to improve the accuracies of models. In addition, increasing different variations of the raw images were also used to improve accuracies. The annotation of images affected the training process. If the boundary is exceeded the model will not continue training.

In the initial stages of training, Google Colab free version was used to train models. It contains 12.69 GB RAM and 78.19 GB Disk. However, when increasing the number of images in the dataset, dimensions and epochs were affected by the training process. Google Colab's free version only provides limited features to the users. Therefore, the Google Colab pro version was purchased, which provides more features such as faster GPUs, longer runtimes (fewer idle runtimes), and more memory (25.46 GB and 116.83GB) as for the solution.

6.3 Discussion

The main objective of this individual research component is to develop a mobile application though which coconut growers can easily identify the coconut caterpillar infestation while differentiating its symptoms from symptoms of other diseases and infestations (leaf scorch disorder). The progression level of the infestation with the number of caterpillars infested is also needed to aid control measures to avoid potential outbreaks. Initially, a background study was conducted in order to properly identify the requirements, design the system, and finalize the technologies and tools required. Due to the lack of knowledge in the domain in the initial phase, many trainings, discussions and readings were done to gain a thorough understanding.

As discussed in the results, YOLOv5 object detection algorithm was used to calculate the number of caterpillars. This version was selected after a comparison with an earlier version (YOLOv3). According to the test results, training loss of YOLOv3 was

comparatively high, and the process was time consuming. The model also struggled in generalizing the objects in new or unusual aspect ratios. As for YOLOv5, since it is written in PyTorch the training process was fast and easy to configure. The trained weights of YOLOv5 was significantly small in size, making it easy for deployment. Prediction rates in YOLOv5 was significantly higher than that of YOLOv3 due to the new "auto-anchor" feature.

The YOLOv5 model was further evaluated using two sub-versions YOLOv5s and YOLOv5x. The detection speed of YOLOv5s is relatively faster than that of YOLOv5x. However, due to its small network structure, the learning rate and accuracy of YOLOv5s is lower than YOLOv5x. As a result, the detection model for this study was trained using YOLOv5x.

When annotating the images, labelling caterpillars was found to be difficult since they hide inside galleries. Their entire body may not be visible. Furthermore, due to the difficulty in annotating all of the leaflets in coconut fronds, a single leaflet was used for mask R-CNN model training. It is best to avoid marking positions outside the boundary when annotating images with VGG annotator. The training process cannot be continued if the coordinates are outside the boundary. Images with lower dimensions may result in poor extraction of essential features, whereas images with higher dimensions require better computational power (higher Graphic Processing Unit (GPU)) to train models. Therefore, images with dimensions of 416 x 416 were used to train the models with the best performance and accuracy.

When deploying the model, a global variable was used to determine whether the model had previously been loaded. If it is loaded, the system uses the model variable without having to load it every time. This made the identification process much faster and efficient.

7. CONCLUSIONS

The main objective of this study was to develop a mobile application to facilitate early detection and prevention of coconut pests and diseases. In addition, if the images are detected as infected, the dispersion visualization map is updated based on the identified location. Furthermore, notifications are sent to the stakeholders who are residing in the identified area. To determine the dispersion factors of the diseases, the weather data is acquired and visualized statistically for CRISL. The outcome will be displayed to end consumers via mobile and web applications.

In this research component, coconut caterpillar identification, classification and progression level determination were achieved using deep learning technologies such as Mask R-CNN, YOLOv5 object detection algorithm and image processing. For the evaluation of models, performance was assessed by comparing the annotated images with the prediction results during the training process by considering the loss values. After training metrics such as precision, recall, F1 score, and Average Precision (AP) was used to assess the performance. Preprocessing and augmentation techniques as well as hyperparameter tuning was used to enhance the accuracy scores of each model. In the Mask R-CNN model, ResNet-101 achieved high detection rates with a mAP value of 95.26%. The trained Darknet-based YOLOv5 model achieved a mAP value of 96.28%.

Furthermore, the overall system ('CocoRemedy') will be expanded to classify other prevailing coconut pests such as whitefly, black beetle and coconut mite. 'CocoRemedy' application will be enhanced as an e-commerce platform where the buyers can purchase and sellers can exhibit their products such as fertilizers, coconut plants and other related products. Finally, the developed models will be integrated to a drone system to identify diseases and infestations by capturing images of the trees without cutting the leaves.

8. REFERENCES

- [1] N. Noguchi, J. Reid, E. Benson, and T. Stombaugh, "Vision Intelligence for an Agricultural Mobile Robot Using a Neural Network," in *Proceedings of 3rd IFAC (International Federation of Automatic Control) CIGR Workshop on Artificial Intelligence in Agriculture, Makuhari, Chiba, Japan, April 24 - 26 1998*, pp.139-144.
- [2] S. Iyer, 'How smart agriculture solution can empower farmers. soft web solutions, 2021. [Online]. Available: <https://www.softwebsolutions.com/resources/plant-diseases-detection-using-iot.html> [Accessed February 26 2021].
- [3] "Economic and Social statistics of Sri LANKA," [Online]. Available: <https://www.cbsl.gov.lk>. [Accessed: 01-Sep-2021].
- [4] S. Ranasinghe, and I. M. S. K. Idrisinghe, "Status of coconut sector development in Sri Lanka," Country Statement in the report of the 55th APCC session/ministerial meeting, 26-30 August 2019, Manila, Philippines.
- [5] A. D. N. T. Kumara, M. Chandrashekharaiyah, S. B. Kandakoor, and A. K. Chakravarthy, "Status and Management of Three Major Insect Pests of Coconut in the Tropics and Subtropics" in *New Horizons in Insect Science: Towards Sustainable Pest Management*, A. K. Chakravarthy Ed. Springer India, April 2015, pp. 359-381.
- [6] L.C.P. Fernando, "Effective integrated pest management (IPM) technology application in pest & disease management in South Asia," in *Proceedings of the XLVI COCOTECH Conference, Colombo, Sri Lanka, July 7-11 2014*. Pp. 70-79.
- [7] R. Miriyagalla, Y.Sam arawickrama, D. Rathnaweera, L. Liyanage, D. Kasthurirathna, D. Nawinna, and J. Wijekoon, "The Effectiveness of Using Machine Learning and Gaussian Plume Model for Plant Disease Dispersion Prediction and Simulation," in *International Conference on Advancements in Computing (ICAC)*, pp. 317-322, 2019.
- [8] M. L. M. Salgado, History and development of the coconut industry of Ceylon, <https://core.ac.uk/download/pdf/52172781.pdf> accessed on 02 10 2021

- [9]. P. A. C. R. Perera, M. P. Hassell, and H. C. J. Godfray, "Population dynamics of the coconut caterpillar, *Opisina arenosella* Walker (Lepidoptera: Xyloryctidae), in Sri Lanka," *COCOS*, vol. 7, pp. 42–57, 1989.
- [10] Goodhands, "Damages by Coconut Caterpillar (*Opisina arenosella*)," goodhands, 25-Aug-2021. [Online]. Available: <https://goodhands.lk/damages-by-coconut-caterpillar-opisina-arenosella/>. [Accessed: 13-Oct-2021].
- [11] Ref.: Chandrika Mohan, C.P. Radhakrishnan Nair, C. Kesavan Nampoothiri and P. Rajan. Leaf-eating caterpillar (*Opisina arenosella*)-induced yield loss in coconut palm. *International Journal of Tropical Insect Science* , Volume 30 , Issue 3 , September 2010 , pp. 132 - 137 DOI: <https://doi.org/10.1017/S174275841000024X>
- [12]. M. L. M. Salgado, History and development of the coconut industry of Ceylon, <https://core.ac.uk/download/pdf/52172781.pdf> accessed on 02 10 2021[Accessed: 13-Oct-2021].
- [13] L. J. Francel, and S. Panigrahi, "Artificial neural network models of wheat leaf wetness," *Agricultural and Forest Meteorology*, vol. 88 no. 1, pp. 57-65, 1997.
- [14] M. Ismail, and Mustikasari, "Intelligent system for tea leaf disease detection," *IPSJ Tech. Rep.* pp. 1-4, 2013.
- [15] F. Hahn, I. Lopez, and G. Hernandez, "Spectral detection and neural network discrimination of *Rhizopus stolonifer* spores on red tomatoes," *Biosystems Engineering*, vol. 89 no. 1, pp. 93-99, 2004.
- [16] K. Y. Huang, "Application of artificial neural network for detecting *Phalaenopsis* seedling diseases using color and texture features," *Computers and Electronics in agriculture*, vol. 57 no. 1, pp. 3-11, 2007.
- [17] G. M. Pasqual, and J. Mansfield, "Development of a prototype expert system for identification and control of insect pests," *Computers and Electronics in Agriculture*, vol. 2 no. 4, pp. 263-276, 1988.
- [18] Y. Cao, C. Zhang, Q. Chen, Y. Li, S. Qi, and L. Tian, "Identification of species and geographical strains of *Sitophilus oryzae* and *Sitophilus zeamais* using the visible/near-infrared hyperspectral imaging technique," *Pest Manag Sci*, vol. 71, pp. 1113–1121, 2015.

- [19] I. Ghosh, and R. K. Samanta, "TEAPEST: An expert system for insect pest management in tea," *Applied Engineering in Agriculture*, vol. 19 no. 5, pp. 619, 2003.
- [20] G. Bhadane, S. Sharma, and V. B. Nerkar, "Early Pest Identification in Agricultural Crops using Image Processing Techniques," *International Journal of Electrical, Electronics and Computer Engineering*, vol. 2, no. 2, pp. 77-82, 2013.
- [21] A. Chandy, "Pest infestation identification in coconut trees using deep learning," *Journal of Artificial Intelligence and Capsule Networks*, vol. 01, no. 01, pp. 10-18, 2019.
- [22]. P. Singh, A. Verma, and J. S. R. Alex, "Disease and pest infection detection in coconut tree through deep learning techniques," *Computers and Electronics in Agriculture*, vol. 182, March 2021, [Online] Available: <https://www.sciencedirect.com/science/article/pii/>, [Accessed 2 February 2021].
- [23] Q. Wang, F. Qi, M. Sun, J. Qu, and J. Xue, "Identification of tomato disease types and detection of infected areas based on deep convolutional neural networks and object detection techniques," *Computational Intelligence and Neuroscience*, 16-Dec-2019. [Online]. Available: <https://www.hindawi.com/journals/cin/2019/9142753/>. [Accessed: 13-Oct-2021].
- [24] S. Manoharan, B. Sariffodeen, K. T. Ramasinghe, L. H. Rajaratne, D. Kasthurirathna, and J. L. Wijekoon, "Smart plant disorder identification using computer vision technology," 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference, IEMCON 2020.
- [25] A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," in *Proceedings of the 18th international Conference on Pattern Recognition (ICPR'06)*, pp. 850–855, IEEE, Hong Kong, China, August 2006.
- [26] E. A. Lins, J. P. M. Rodriguez, S. I. Scoloski, J. Pivato, M. B. Lima, J. M. C. Fernandes, P. R. Valle da Silva Pereira, D. Lau, and R. Rieder, "A method for counting and classifying aphids using computer vision," *Computers and Electronics in Agriculture*, vol. 169, February 2020, [Online] Available: <https://www.canva.com/design/DAEXiXDvMLI/YbsWu2jO2RiqiBKvFM6-Yg/edit>. [Accessed 8 February 2021].




- [27]. U. B. M. Ekanayake, "Leaf scorch decline of coconut," *Ceylon Cocon. Quart.* vol. 19, pp. 183-187, 1968.
- [28] I. A. Alshawwa, A. A. Elsharif, and S. S. Abu-Naser, "An Expert System for Coconut Diseases Diagnosis" *International Journal of Academic Engineering Research (IJAER)*, vol. 3, no. 4, pp. 8-13, April 2019.
- [29] T. Silva, "Image panorama stitching with opencv," Medium, 02-Aug-2019. [Online]. Available: <https://towardsdatascience.com/image-panorama-stitching-with-opencv-2402bde6b46c>. [Accessed: 13-Oct-2021].
- [30] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN." [openaccess.thecvf.com](https://openaccess.thecvf.com/content_ICCV_2017/papers/He_Mask_RCNN_ICCV_2017_paper.pdf), 2017. Available: https://openaccess.thecvf.com/content_ICCV_2017/papers/He_Mask_RCNN_ICCV_2017_paper.pdf
- [31] Y. Yu, K. Zhang, L. Yang, and D. Zhang, "Fruit detection for strawberry harvesting robot IN non-structural environment based ON Mask-RCNN," *Computers and Electronics in Agriculture*, vol. 163, p. 104846, 2019.
- [32] Y. Qiao, M. Truman, and S. Sukkarieh, "Cattle segmentation and contour extraction based on mask R-CNN for Precision Livestock Farming," *Computers and Electronics in Agriculture*, vol. 165, p. 104958, 2019.
- [33] P. D. R. Matthew Stewart, "Simple introduction to Convolutional Neural Networks," Medium, 29-Jul-2020. [Online]. Available: <https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac>. [Accessed: 13-Oct-2021].
- [34] A. Rahman and Y. Wang, "Optimizing intersection-overunion in deep neural networks for image segmentation," in *Proceedings of the International Symposium on Visual Computing*, pp. 234–244, Springer, Las Vegas, NV, USA, December 2016.
- [35] J. Salau and J. Krieter, "Instance segmentation with Mask R-CNN applied to loose-housed dairy cows in a multi-camera setting" *Animals*, vol. 10, pp. 2402; 2020 doi:10.3390/ani10122402
- [36] "Remote Sensing," An Open Access Journal from MDPI. [Online]. Available: <https://www.mdpi.com/journal/remotesensing>. [Accessed: 13-Oct-2021].

- [37] E. Sun, “Object extraction based on instance segmentation,” LinkedIn, 27-Nov-2020. [Online]. Available: <https://www.linkedin.com/pulse/object-extraction-based-instance-segmentation-evelyn-sun>. [Accessed: 13-Oct-2021].
- [38] - J. Strickland, “K-means clustering using R,” BI Corner, 18-Jun-2015. [Online]. Available: <https://bicorner.com/2015/05/26/k-means-clustering-using-r/>. [Accessed: 13-Oct-2021].
- [39] “Learn by marketing,” Learn by Marketing | Data Mining + Marketing in Plain English. [Online]. Available: <https://www.learnbymarketing.com/methods/k-means-clustering/>. [Accessed: 13-Oct-2021].
- [40] “Edge detection with Gaussian Blur,” Rhea. [Online]. Available: https://www.projectrhea.org/rhea/index.php/Edge_Detection_with_Gaussian_Blur. [Accessed: 13-Oct-2021].
- [41] S. Banerjee, “Counting grains of rice-corona quarantine blues,” Medium, 30-Mar-2020. [Online]. Available: <https://medium.com/@sumandeep.banerjee/counting-grains-of-rice-corona-quarantine-blues-44eaba6d3598>. [Accessed: 13-Oct-2021].
- [42] “Canny edge detection¶,” OpenCV. [Online]. Available: https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html. [Accessed: 13-Oct-2021].
- [43] “Morphological transformations¶,” OpenCV. [Online]. Available: https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html. [Accessed: 13-Oct-2021].
- [45] J. Brownlee, “A gentle introduction to object recognition with deep learning,” Machine Learning Mastery, 26-Jan-2021. [Online]. Available: <https://machinelearningmastery.com/object-recognition-with-deep-learning/>. [Accessed: 13-Oct-2021].
- [46] - “Make sense,” Make Sense. [Online]. Available: <https://www.makesense.ai/>. [Accessed: 13-Oct-2021].

- [47] Ultralytics, “Ultralytics/yolov5: Yolov5 in PyTorch & ONNX & CoreML & TFLite,” GitHub. [Online]. Available: <https://github.com/ultralytics/yolov5>. [Accessed: 13-Oct-2021].
- [48] Y. Chen, C. Zhang, T. Qiao, J. Xiong, B. Liu, "Ship detection in optical sensing images based on YOLOv5," in SPIE 11720, *Proceedings of the Twelfth International Conference on Graphics and Image Processing (ICGIP2020)*, Xi'an, China January 27, 2021; doi: 10.1117/12.2589395
- [49] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “Yolov4: Optimal Speed and accuracy of object detection,” arXiv.org, 23-Apr-2020. [Online]. Available: <https://arxiv.org/abs/2004.10934v1>. [Accessed: 13-Oct-2021].
- [50] C-Y. Wang, H-Y. M. Liao, I-H. Yeh, Y-H Wu, P-Y Chen, and J-W Hsieh, “CSPNET: a new backbone that can enhance learning capability of CNN” arXiv:1911.11929v1 [cs.CV] 27 Nov 2019
- [51] R. Xu, H. Lin, K. Lu, L. Cao and Y. Liu, “A forest fire detection system based on ensemble learning”, *Forests*, vol 12, pp 217, 2021, <https://doi.org/10.3390/f12020217>
- [52] “Give your software the power to see objects in images and video,” Roboflow. [Online]. Available: <https://roboflow.com/>. [Accessed: 13-Oct-2021].
- [53] “You only look once: Unified, real-time object detection ...” [Online]. Available: https://www.researchgate.net/publication/278049038_You_Only_Look_Once_Unified_Real-Time_Object_Detection. [Accessed: 13-Oct-2021].
- [54] T-L. Lin, H-Y. Chang, and K-H. Chen, “The pest and disease identification in the growth of sweet peppers using Faster R-CNN and Mask R-CNN” *Journal of Internet Technology* Vol. 21 No.2, pp 605-614, 2020

9. APPENDICES

Appendix - A : Plagiarism report

Research Project Final Report		Start 05-Oct-2021 10:48AM Due 31-Dec-2021 11:59PM Post 13-Oct-2021 12:00AM	14% 	Resubmit View 
-------------------------------	---	--	---	---

Appendix - B : Sample Questionnaire

<https://forms.gle/pXNQrAichCegcovi8>