

# **AI ASISSTANT CHATBOT FOR LOAN ELIGIBILITY PREDICTION SYSTEM**

Hilma M.I.F

IT21142178

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

Department of Computer Science and Software Engineering

Sri Lanka Institute of Information Technology  
Sri Lanka

May 2025

## 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
Hilma M.I. F	IT21142178	

Signature of the Supervisor  
(Ms. Suranjini)

Date

.....

.....

## ABSTRACT

In the digital transformation era, banking institutions are increasingly integrating artificial intelligence (AI) to improve operational efficiency and customer service. Ceylan Bank, like many financial organizations, faces growing demands to fast track loan related inquiries while maintaining personalized service and regulatory compliance. Traditional customer service channels, including call centers and in person consultations, are often limited by availability, scalability, and human error. This study proposes the design and development of an AI powered, role based intelligent chatbot system designed to Ceylan Bank's operational needs. The solution utilizes natural language processing (NLP), integrated with a custom built knowledge base, Firebase Authentication, and SQL databases to deliver intelligent, real-time responses to loan related queries. The system is structured around three distinct user roles: general users, existing customers, and bank staff, each with customized access and functionality. General users can inquire about loan types, eligibility, and required documents using natural language. Authenticated customers can access their personalized loan data, including application status and repayment information, retrieved securely from a SQL database. Authorized bank staff are performed with advanced chatbot tools to perform instant, role specific queries on the entire loan system, significantly reducing administrative overhead. The chatbot backend employs Retrieval-Augmented Generation (RAG) to provide a hybrid of generative flexibility and factual accuracy. The AI Chatbot solution aims to improve customer engagement, reduce staff workload, and improve decision making through intelligent automation. Performance evaluations are conducted via user feedback, task accuracy, and system latency metrics to validate the chatbot's effectiveness across all roles.

**Keywords** - AI Chatbot, Loan Management, Role-Based Access, Natural Language Processing, Firebase Authentication, SQL Integration, Financial Technology

## **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
<b>3.1 Main Objectives .....</b>	<b>15</b>
<b>3.2 Specific Objectives .....</b>	<b>15</b>
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
NLP	Coconut Caterpillar Infestation
AI	Artificial Intelligence
RAG	retrieval-augmented generation
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**

In today's rapidly evolving financial landscape, the demand for intelligent, efficient, and user centric solutions has intensified, particularly in the domain of loan eligibility and management. Traditional loan processing systems often involve time consuming procedures, manual document verification, and delayed responses resulting in inefficiencies for both customers and bank staff. As financial institutions strive to modernize operations and improve customer satisfaction, the integration of AI technologies has become not only beneficial but essential.

This research focuses on the design and development of an AI-powered chatbot system to improve the loan eligibility process within banking environments. The proposed system introduces a smart, role-based chatbot assistant that serves three primary user groups: general users, existing customers, and bank staff. Through natural language understanding and intelligent data retrieval, the chatbot delivers relevant information and services in a conversational format, improving user interactions and reducing operational overhead.

By using NLP, RAG, a structured knowledge base, and secure database integration, the chatbot offers real time support and tailored responses. General users are provided with detailed guidance on loan eligibility requirements and required documentation. Logged-in existing customers can inquire about their loan status, repayment schedules, and financial obligations. Bank staff can directly retrieve customer loan details without manual database queries, significantly improving service efficiency.

This report presents the development and implementation of this AI chatbot system, focusing on its architecture, role-specific functionalities, and real world use cases. The goal of this project is not only to automate routine financial interactions but to also set a precedent for AI driven personalization and operational optimization in emerging markets.

## **1.2 Background Literature**

### **1.2.1 Evolution of AI Chatbots in Financial Services**

Artificial Intelligence has revolutionized numerous industries, with the banking sector being among the most significantly impacted. Chatbots, a subset of AI powered by Natural Language Processing, have emerged as intelligent interfaces that improves customer service, automate queries, and personalize banking experiences. Over the past decade, leading global banks have adopted AI-powered virtual assistants to improve customer satisfaction and reduce human resource costs.

The primary objective of AI chatbots in banking is to deliver instant, accurate responses to customer queries, ranging from basic FAQs to complex tasks. Early implementations were limited to rule based decision trees and keyword detection. However, modern advancements such as Transformer based NLP models (e.g., BERT, GPT) and Retrieval-Augmented Generation have significantly improved the contextual understanding and dynamic response capabilities of chatbots

### **1.2.2 Retrieval Augmented Generation for Knowledge-Based Chatbots**

The integration of RAG in chatbot systems is a recent breakthrough improving the accuracy and contextual richness of responses. Unlike traditional generative models, which rely solely on learned parameters, RAG uses external knowledge bases by retrieving relevant documents and incorporating them into the generation process. This is particularly valuable in domains like banking where precise regulatory, financial, and procedural data are necessary.

For example, in loan related queries, RAG allows chatbots to cite up to date information from bank policies, loan agreements, and eligibility requirements. This blend of retrieval and generation allows that chatbot responses are both contextually relevant and factually grounded which is a critical need in financial services where accuracy is non negotiable.

### **1.2.3 AI Chatbots in Sri Lankan Banking Context**

In Sri Lanka, AI adoption in banking is still at an emerging stage, with few banks experimenting with smart assistants. However, digital transformation is accelerating. Ceylan Bank, recognizing the importance of customer centric innovation, initiated the development of a multi-user AI chatbot customed to provide general and personalized loan related support. Unlike generic bots, this solution dynamically interacts with both public facing knowledge (e.g., required documentation, policies) and private user specific data (e.g., repayment status, loan amounts) securely retrieved from databases.

This system bridges the gap between traditional customer service and modern AI powered assistance, providing value to three distinct user groups: general visitors, existing loan customers, and bank staff. Although the use of AI chatbots in local banks is limited compared to global institutions, the success of this project could serve as a model for broader adoption across Sri Lanka's financial sector.

## **1.2 Research Gap**

Despite the growing adoption of AI-based chatbots in the global financial sector, several gaps remain particularly in the Sri Lankan context and in the integration of advanced retrieval-generation mechanisms in banking AI assistants.

First, most existing chatbot systems in banks are designed for general inquiries and offer limited personalized responses. These bots typically rely on rule based architectures or simple NLP models that cannot retrieve or interpret customer specific data in real time. As a result, they fail to meet the evolving expectations of users who demand instant, accurate, and context aware assistance, especially in sensitive domains such as loan management.

Second, there is a noticeable absence of multi role chatbot systems developed for different user groups such as general visitors, existing customers, and internal staff. Most commercial solutions are either customer-facing or internal tools but rarely both. This one-size-fits-all approach leads to inefficient information delivery and poor user experience.

Third, the use of RAG models, which combine contextual knowledge retrieval with dynamic response generation, is still under explored in local banking applications. While RAG has shown promise in academic and enterprise domains, its implementation in high stakes, multi-user financial systems like loan management remains sparse.

The below Table 1.1 shows a tabularized format of comparative analysis of existing systems vs. proposed solution

Table 1. 1: comparative analysis of existing systems vs. proposed solution

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	✓	✓	✓	✓

## 2. RESEARCH PROBLEM

In the rapidly evolving digital banking landscape, customer expectations for instant, intelligent, and context-aware support have significantly increased. Traditional banking systems and helpline services are often unable to meet these expectations due to limitations in scalability, personalization, and responsiveness. This is particularly evident in the domain of loan management, where general inquiries, personalized loan status requests, and internal data retrieval by staff demand different levels of access, security, and intelligence.

Currently, Ceylan Bank does not possess an AI-powered, multi-role chatbot system capable of addressing these challenges in an integrated manner. Customers seeking information on loan eligibility, document requirements, or status updates must rely on manual processes such as branch visits or long response times from service agents. Similarly, bank staff members are required to navigate internal databases to retrieve customer information, increasing operational workload and introducing possibilities for human error or delay.

While many international banks have begun deploying chatbot technologies, these systems are typically limited to FAQ-style answers or predefined response flows. They often lack real-time integration with back-end databases and do not support role-based access control. More critically, there is no known implementation within Sri Lankan banks that utilizes advanced RAG methods powered by GPT, paired with real-time authentication and SQL data access tailored for both customers and internal staff.

This research addresses the critical gap by developing an AI chatbot tailored for Ceylan Bank, capable of serving three distinct user groups:

- **General users:** provides loan related information from a curated knowledge base.
- **Existing customers:** Can retrieve their specific loan eligibility and application details.
- **Bank staff:** Efficiently query customer records and loan data without direct database access.



By combining Firebase authentication, SQL data retrieval, and NLP, the proposed solution aims to bridge the gap between user expectations and system capabilities, thus enhancing operational efficiency and customer satisfaction in a secure, scalable, and intelligent manner

### **3. RESEARCH OBJECTIVES**

#### **3.1 Main Objectives**

The main objective of this research is to develop an AI powered chatbot system capable of facilitating intelligent, role-specific communication between users and Ceylan Bank's loan services. The chatbot will support three distinct user groups such as general users, existing customers, and bank staff by using NLP, role based access, and database connectivity to provide accurate, context-aware responses to loan-related queries. This system focuses on improving the efficiency, accessibility, and responsiveness of the bank's customer service while also reducing staff workload and minimizing delays in loan processing and inquiry handling.

#### **3.2 Specific Objectives**

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

##### **Develop a general-purpose chatbot interface for non-authenticated users**

- Integrate a knowledge base trained on Ceylan Bank's loan policies, eligibility requirements, and document guidelines. Use GPT-based NLP to interpret user intent and provide human-like responses.

##### **Implement a personalized chatbot for authenticated existing customers**

- Use of Firebase Authentication to validate and identify users securely and allow customers to retrieve their personal loan details, including loan status, repayment schedules and loan amounts and then Integrate with a SQL database to fetch realtime loan information linked to the authenticated existing customers

### **Design an internal chatbot tool for authorized bank staff**

- Authorized Bank Staff should be able to query the entire loan database securely through natural language prompts and support advanced search and filtering (e.g., customer name, NIC, loan ID, overdue accounts) to ease internal tasks and make sure that access is controlled and monitored for data privacy and compliance

## 4. METHODOLOGY

### 4.1 System Overview

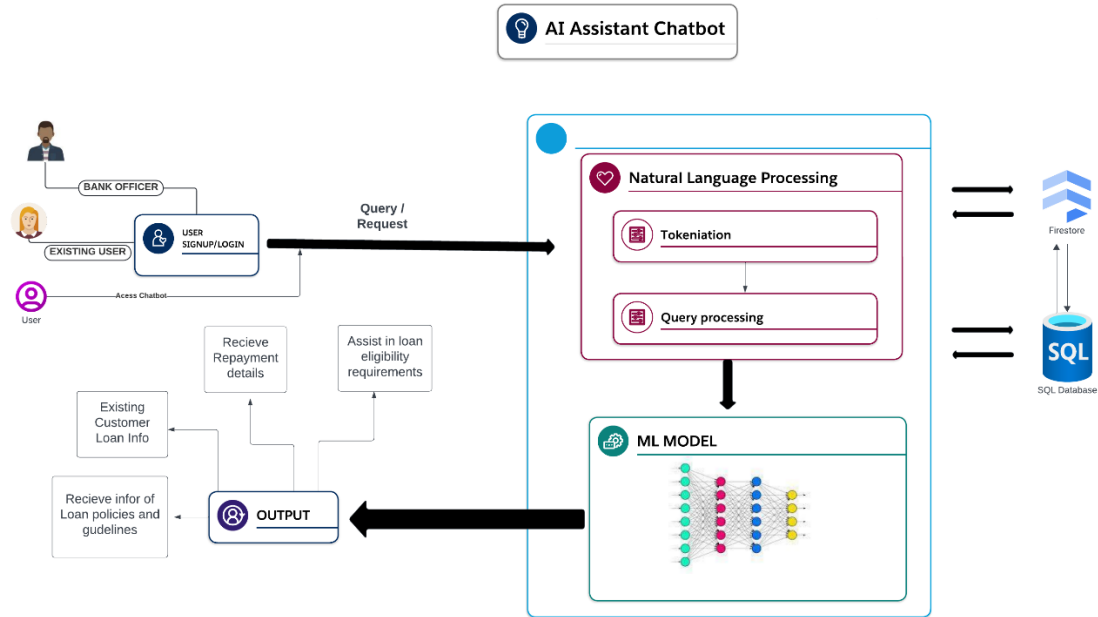


Figure 4. 1: Overview system diagram

Fig 4. illustrates the overall system diagram of the proposed intelligent chatbot solution designed to improve the loan related query experience for different stakeholders of Ceylan Bank. The system serves three primary user roles: general users, existing customers, and bank officers, each with a specific set of privileges and access to information.

As shown in the diagram, all users initiate communication by submitting a query to the chatbot interface. These queries, typed in natural language, are forwarded to the backend. Once received, the queries are passed through a tokenization and preprocessing pipeline where they are normalized and classified.

Depending on the user role, the system follows three distinct paths:

- General Users: Queries are matched against a custom built knowledge base containing predefined answers to general loan questions, eligibility criteria, required documents, and interest rates. No login, authentication or data access is

required for these users.

- Existing Customers: After successful authentication via Firebase, user-specific queries (e.g., “What is the status of my loan application?”) are processed by generating secure SQL queries to fetch real-time loan data from the relational database. The retrieved results are formatted using the NLP model to produce a natural sounding response.
- Bank Staff: These users are authenticated through a separate Firebase role and can query any customer's data across the system. Their queries are similarly interpreted and converted into broader SQL statements, providing full access to the underlying database for operational and administrative insights.

After identifying the query type and required data access level, the system connects to either the custom knowledge base or the SQL loan database. Responses are synthesized by the integrated NLP engine, which combines both the retrieved content and user context to generate a coherent reply. The final output is delivered back to the user via the chatbot UI.

In addition to its conversational functions, the system provides data security and role based access by using Firebase's user management features and SQL parameterization to prevent unauthorized access or SQL injection.

The proposed solution consists of the following key components:

1. Tokenization and Intent Classification

Natural language inputs are parsed, tokenized, and classified to understand user intent and determine whether a response requires knowledge base access or dynamic data retrieval.

2. Firebase Authentication and Role Management

Users are authenticated using Firebase to determine their access role either as a general user, an existing customer, or a staff member and make sure that only authorized users can access sensitive loan data.

### 3. Data Layer Integration with SQL Database

For existing customers and bank staff, the system generates dynamic SQL queries based on the user's intent and securely fetches relevant loan records, such as approval status, repayment schedules, or due dates.

### 4. Response Generation via NLP Model

Once the appropriate data or knowledge is retrieved, the query and result are sent to the NLP model, which formulates the final response in natural language, giving clarity and contextual correctness.

#### 4.1.1 Custom Knowledge Base Functionality

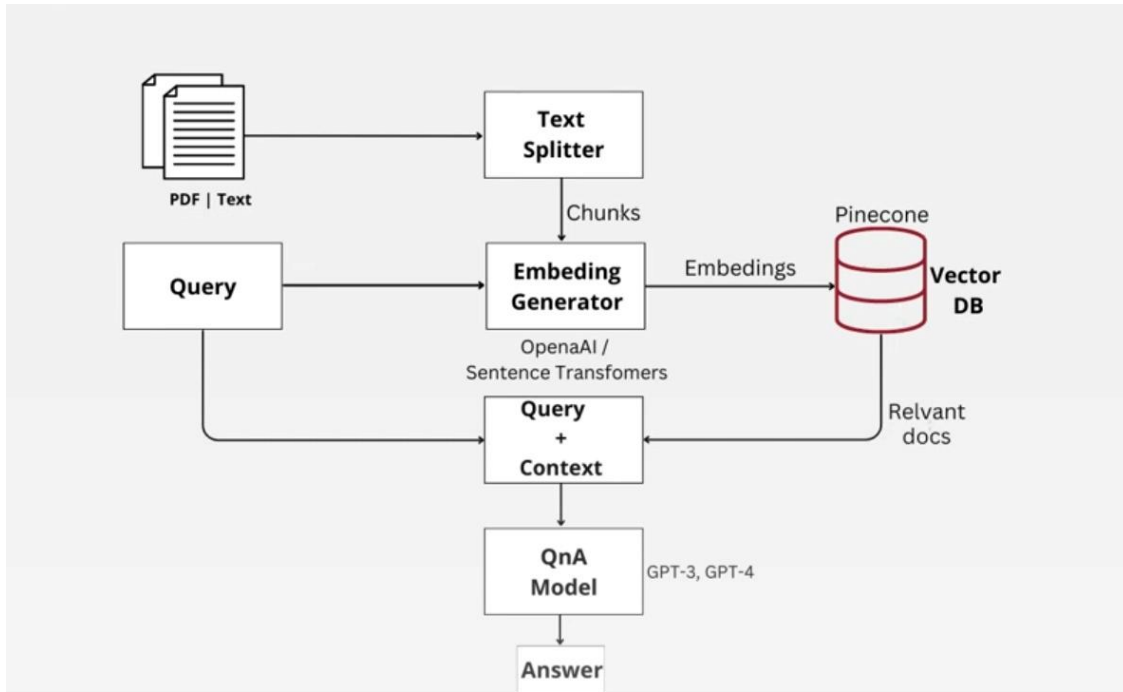


Figure 4. 2: Overview of Custom Knowledge base

The custom knowledge base component of the proposed chatbot system is designed to handle general loan related queries posed by unauthenticated users. This component plays a important role in making sure that frequently asked questions FAQs, policy guidelines, and eligibility criteria are accessible in a conversational format without requiring access to sensitive customer data. The knowledge base integrates RAG architecture to combine the advantages of both semantic document retrieval and generative response synthesis.

Figure 4.2 illustrates the internal workings of this knowledge base, which is divided into main stages such as Data Ingestion and Preparation, Embedding Generation and Storage, Query Processing and Retrieval, and Answer Generation.

##### 1. Data Ingestion and Preparation

The knowledge base is populated with raw textual data sourced from various official documents, such as PDF manuals, financial product brochures, terms and conditions, and internal policy documents. Since these documents tend to be lengthy and structurally

complex, they are first passed through a text splitting algorithm. The Text Splitter divides each document into smaller, semantically meaningful chunks to ensure that the language model can process and retrieve relevant information efficiently. These chunks form the fundamental units used for downstream processing.

## 2. Embedding Generation and Storage

Once the documents have been chunked, each segment is transformed into a high-dimensional numerical vector using a pretrained embedding model. In the implemented solution, either OpenAI's text-embedding-ada-002 model or a Sentence Transformer is used for this purpose. These embeddings capture the semantic essence of each chunk such that similar meanings are mapped close together in vector space.

The resulting vectors are then stored in a Vector Database, such as Pinecone, which is specifically optimized for performing Approximate Nearest Neighbor (ANN) searches in high-dimensional space. Unlike traditional databases that perform keyword searches, the vector database enables fast and efficient semantic similarity retrieval—which is key for locating the most relevant knowledge base entries.

## 3. Query Processing and Retrieval

When a general user submits a natural language question through the chatbot interface, the system processes the query using the same embedding generator that was used for the document chunks. This ensures that both document chunks and user queries reside in the same semantic space, allowing meaningful comparisons.

The query embedding is then used to search the Vector Database, which returns a set of document chunks that are most semantically similar to the query. These retrieved text segments represent the contextual information required to formulate a relevant and accurate answer.

## 4. Answer Generation via LLM

The final stage of the pipeline is handled by a Large Language Model (LLM), such as GPT-3.5 or GPT-4. The model receives both the user's original query and the retrieved



contextual chunks as input. This architecture enables the LLM to generate informed, context-aware responses while avoiding hallucination—an issue common in purely generative models.

#### **4.1.2 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

#### **4.1.3 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.

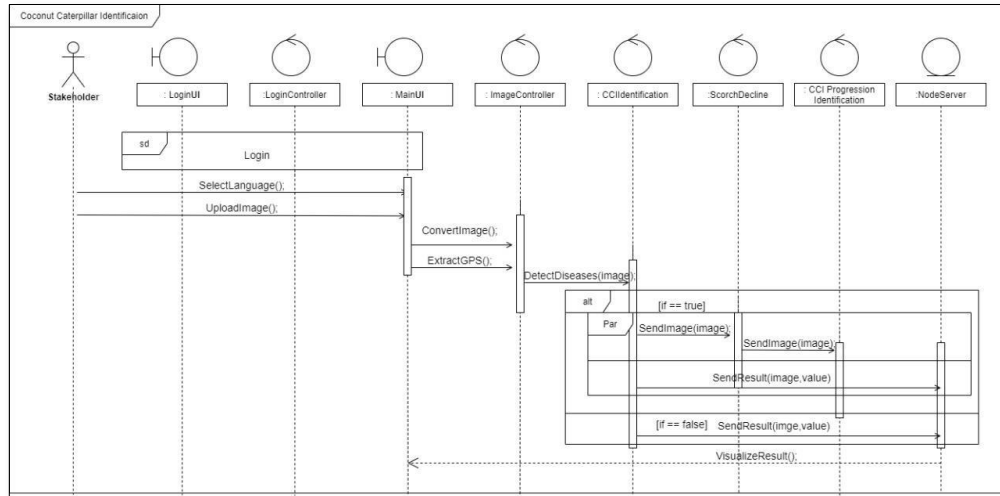


Figure 4. 20: Sequence diagram of CCI component

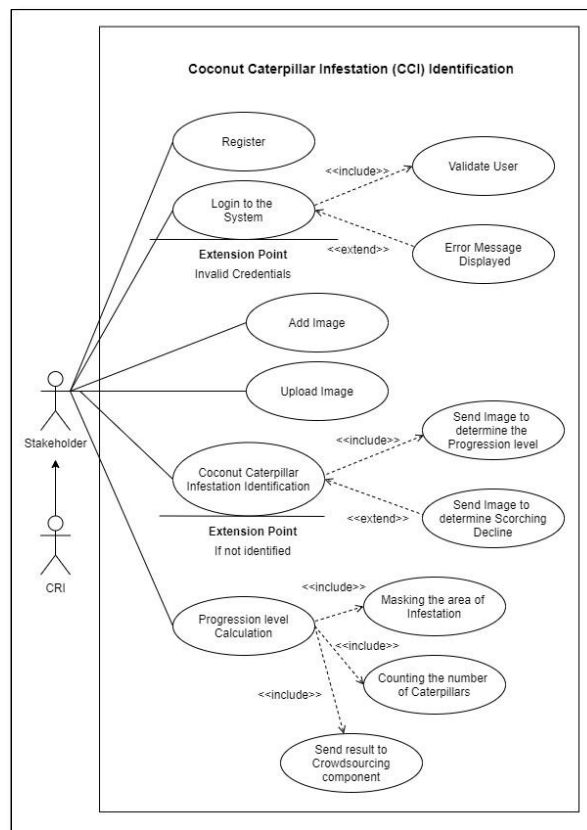


Figure 4. 21: Sequence diagram of CCI component

## **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.

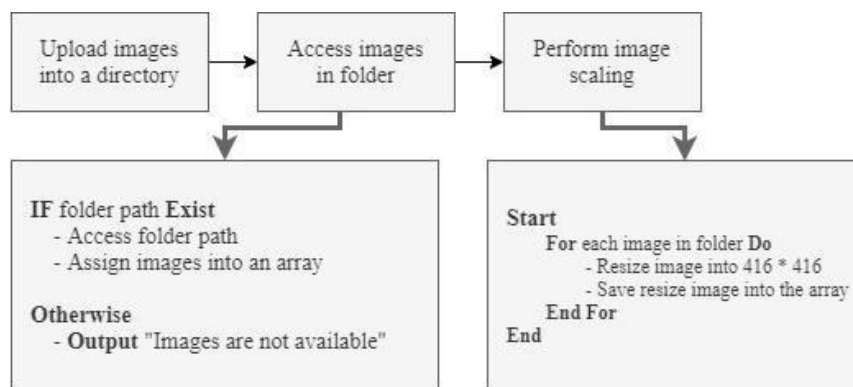


Figure 5. 1: Steps of image preprocessing

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.

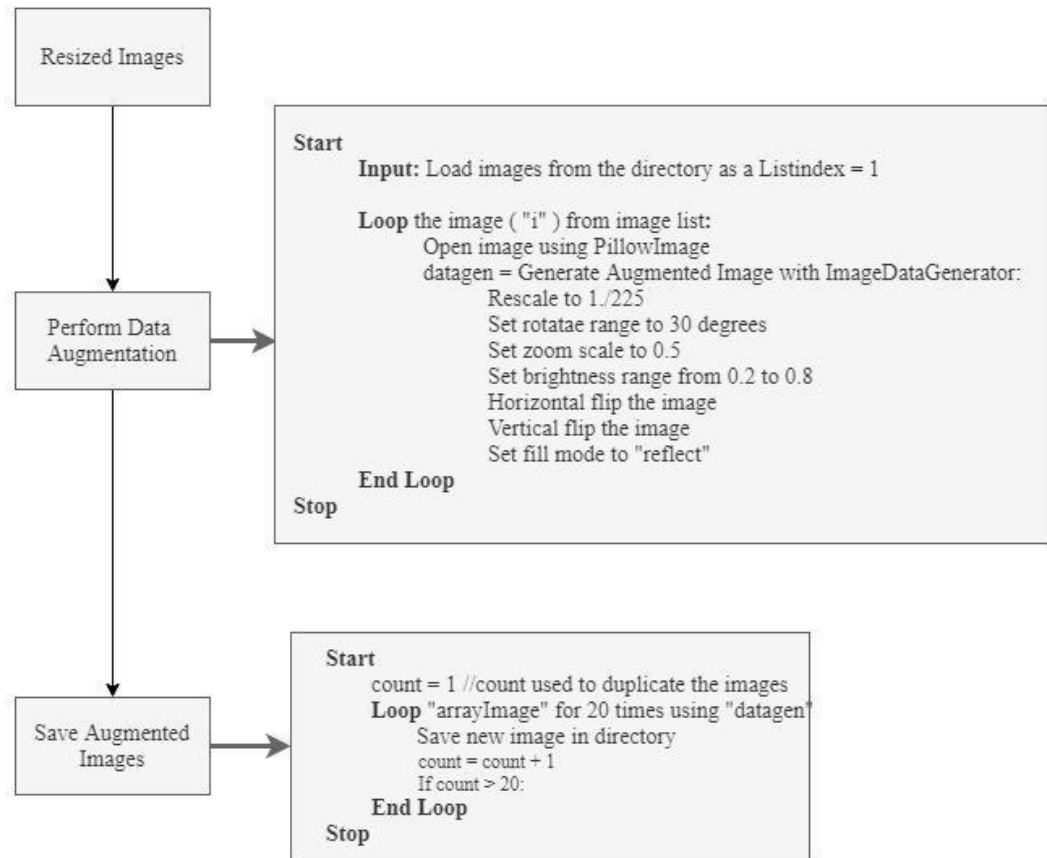


Figure 5. 3: Steps of data augmentation

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

    #perform the augmentation
    arrImg = img_to_array(pillowImg)
    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).

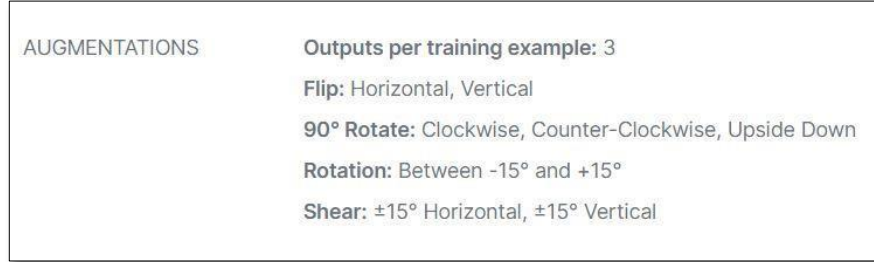


Figure 5. 6: Augmentation techniques applied using Roboflow

### 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.

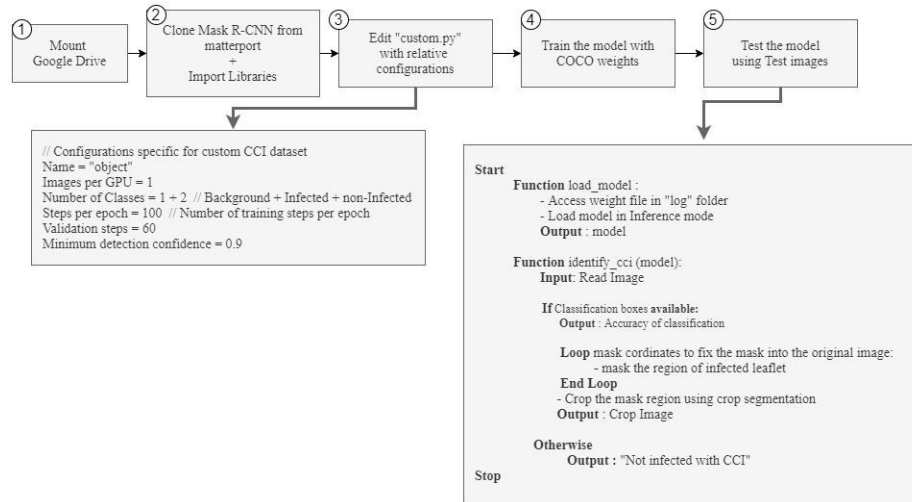


Figure 5. 7: Steps of CCI identification and classification



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<sup>①</sup>(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<sup>③</sup>accordingly (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 code ④ snippet 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<sup>⑤</sup>

```
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.

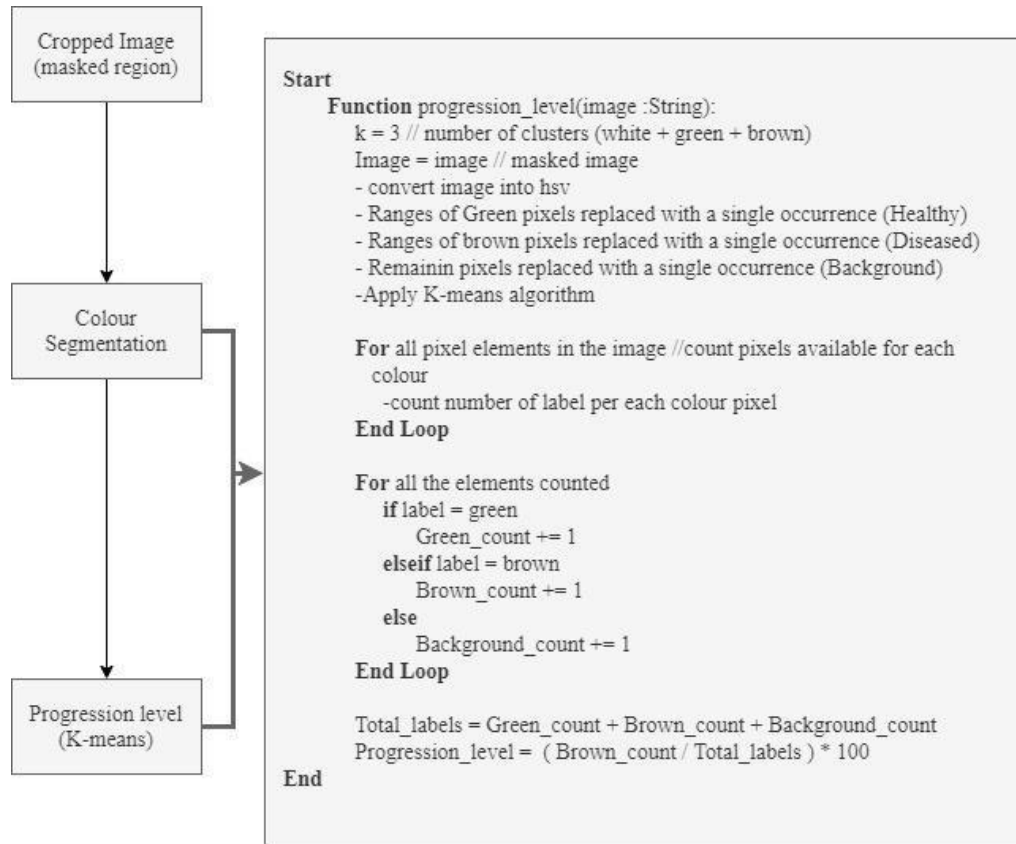


Figure 5. 14: Overview of progression level calculation

- **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.

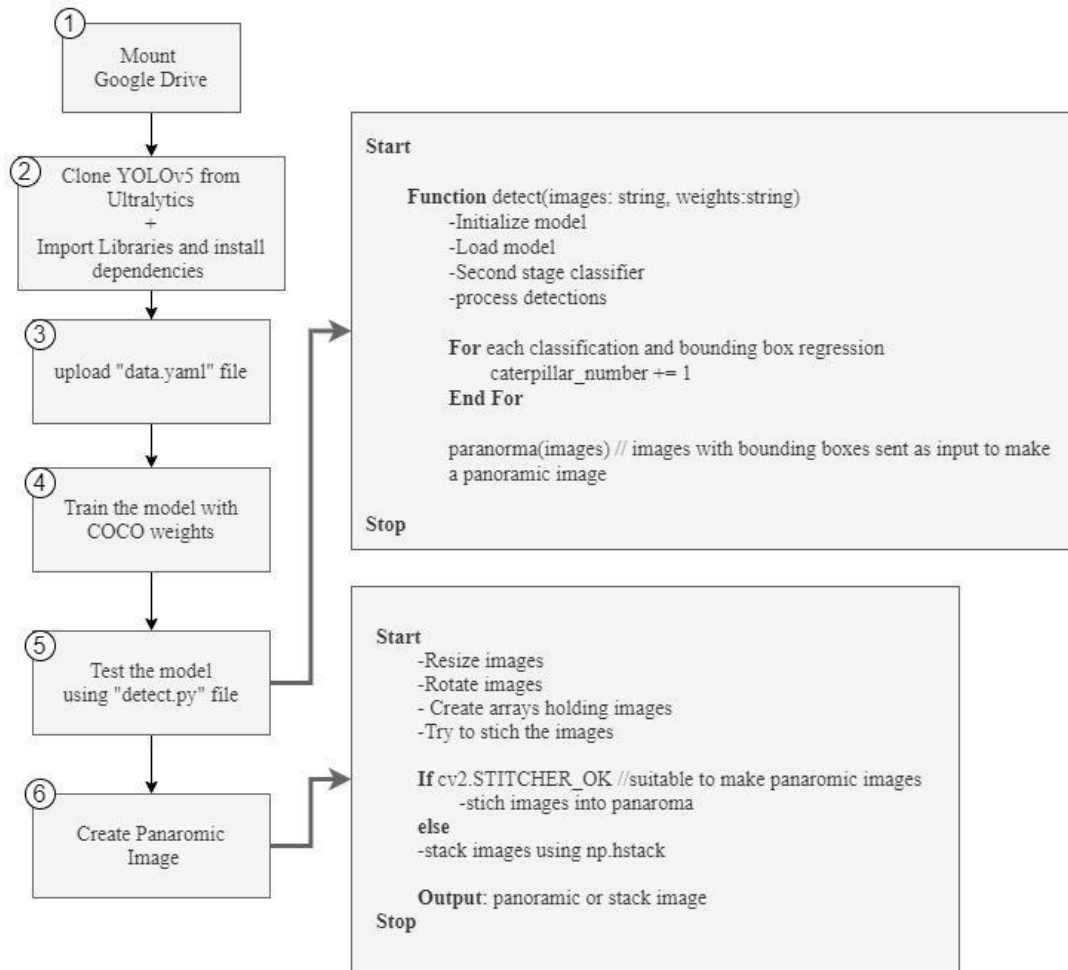
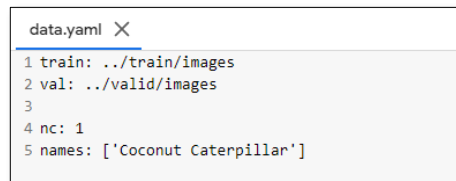


Figure 5. 15: Overview of caterpillar calculation using YOLOv5

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

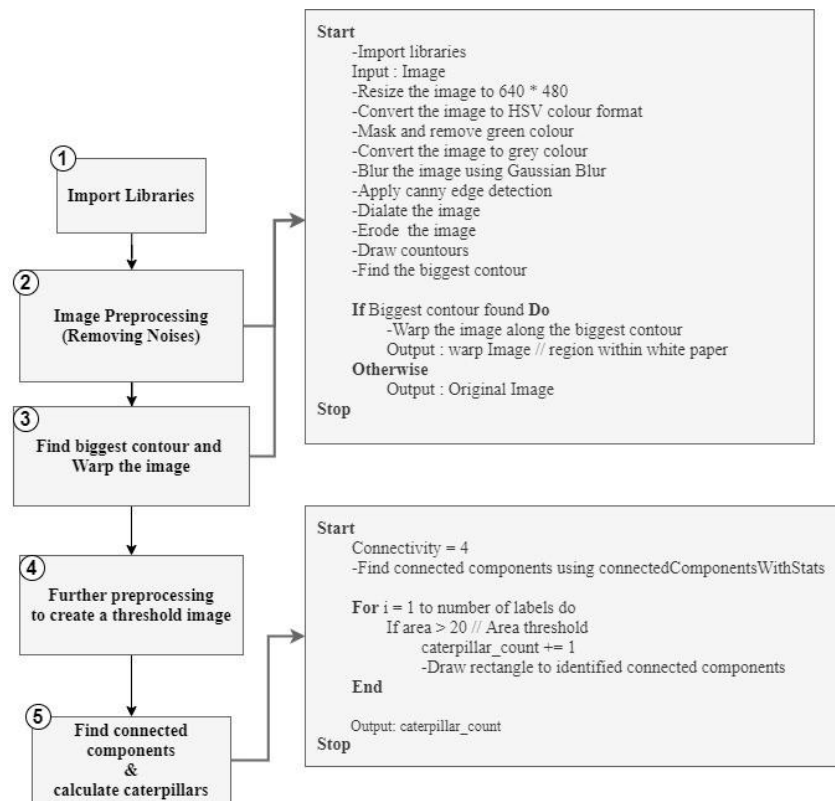


Figure 5. 18: Overview of caterpillar calculation image processing

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).

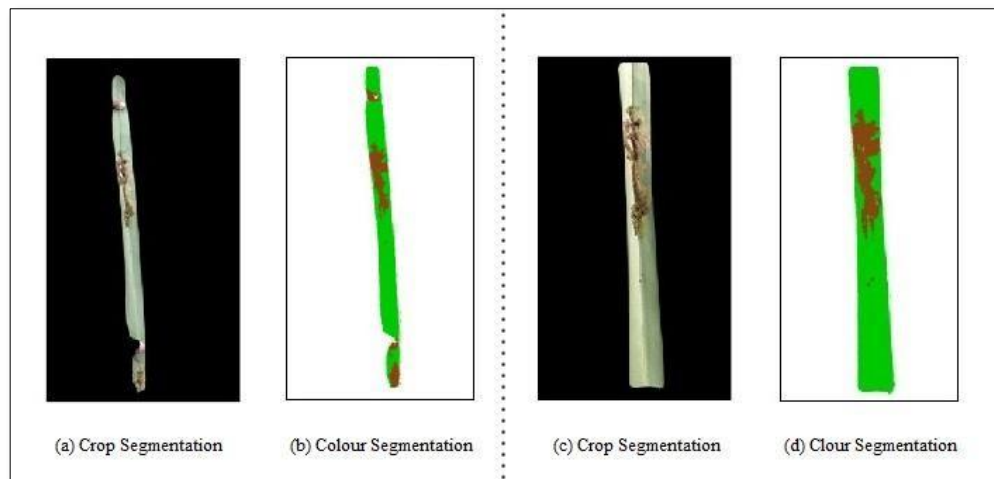


Figure 6. 3: Results of colour segmentation

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.

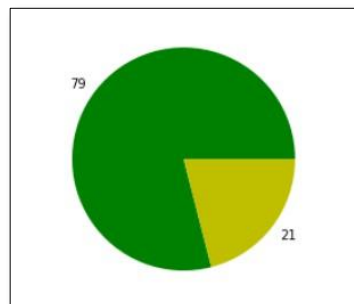


Figure 6. 4: Results of progression level calculation

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

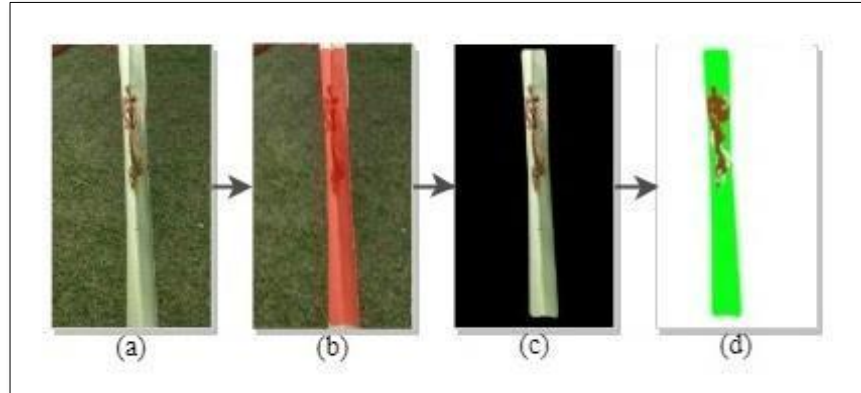


Figure 6. 5: Summarized result of CCI classification and progression level calculation

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.

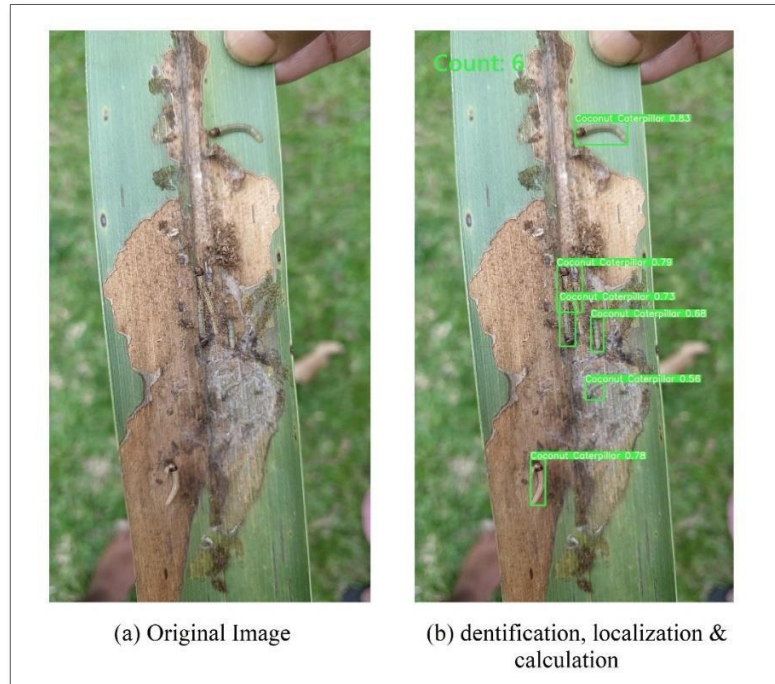


Figure 6.6: Results of caterpillar calculation using YOLOv5

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).

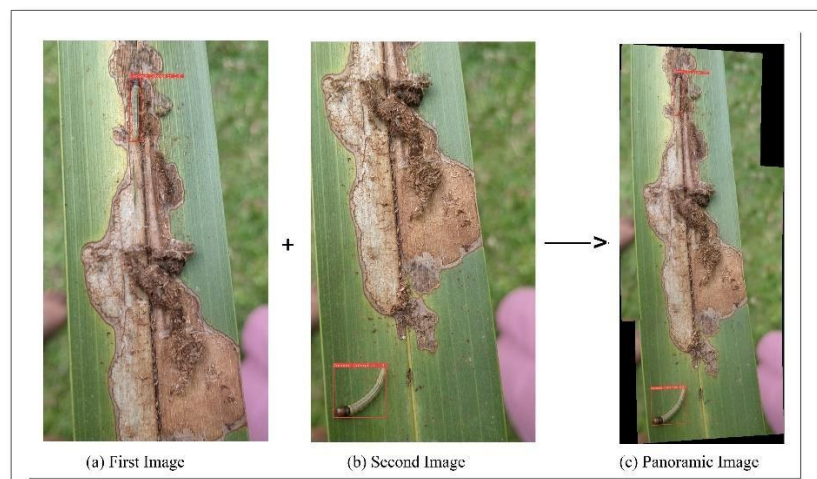


Figure 6.7: Results of creating panoramic image

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

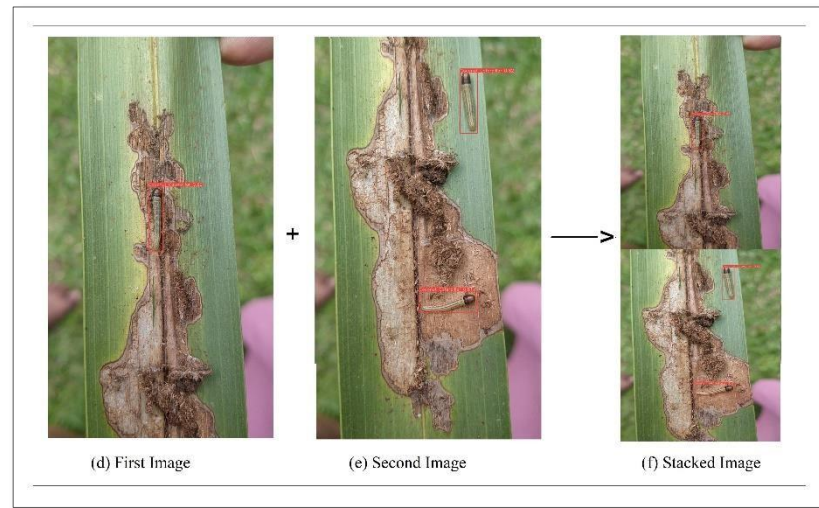


Figure 6. 8: Results of creating stacked image

#### 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.

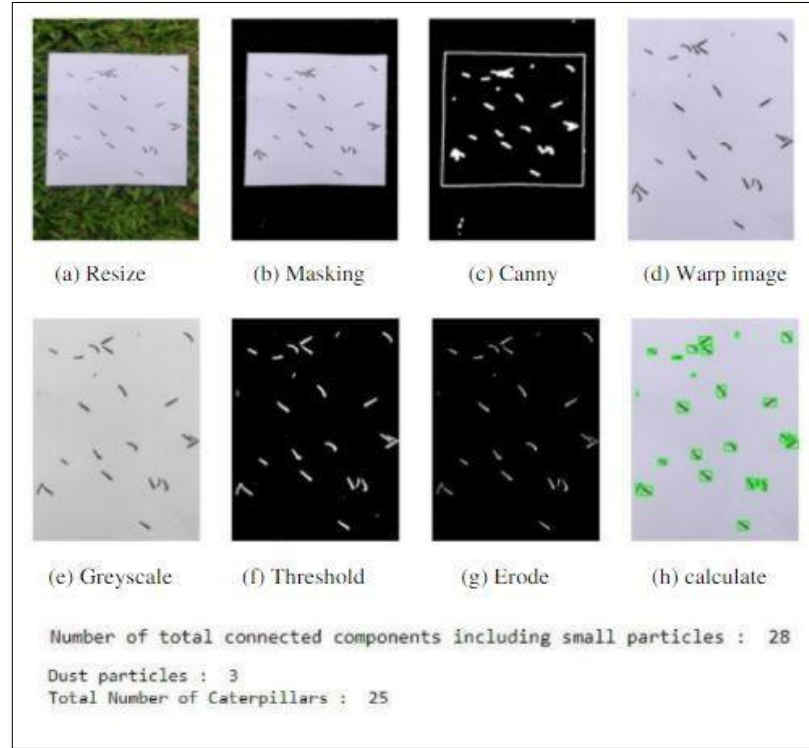


Figure 6. 9: Results of calculating caterpillars using image processing

#### 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

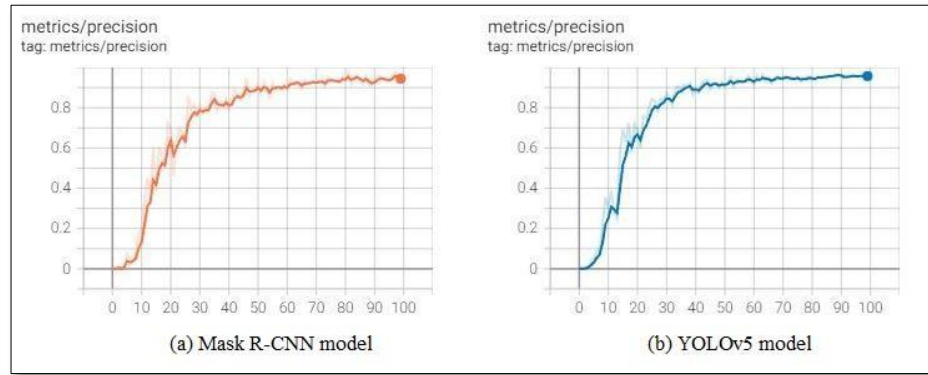


Figure 6. 10: Precision graphs for Mask R-CNN and YOLOv5 models

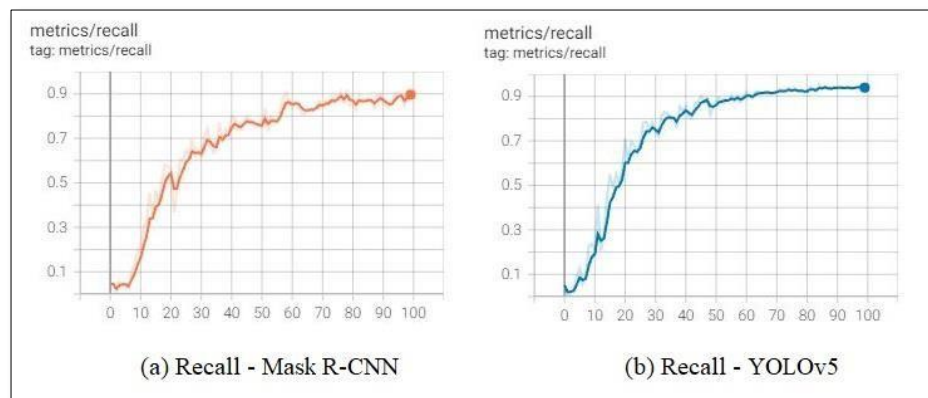


Figure 6. 11: Recall graphs for Mask R-CNN and YOLOv5 models

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.

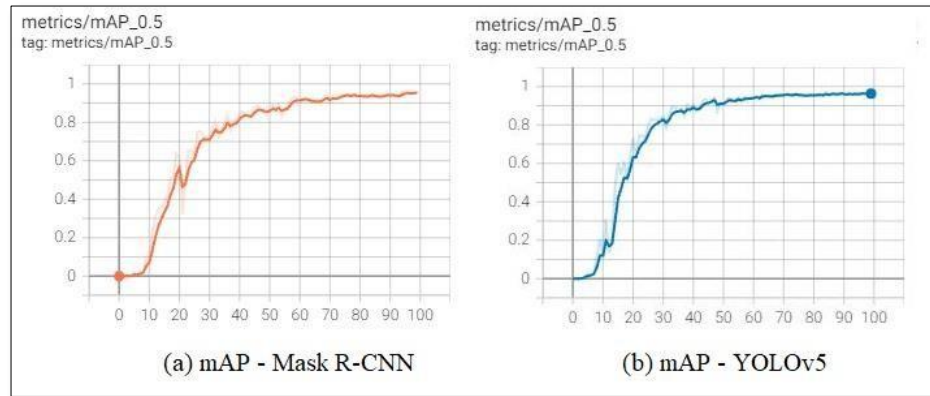


Figure 6. 12: mAP graphs for Mask R-CNN and YOLOv5 models



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.

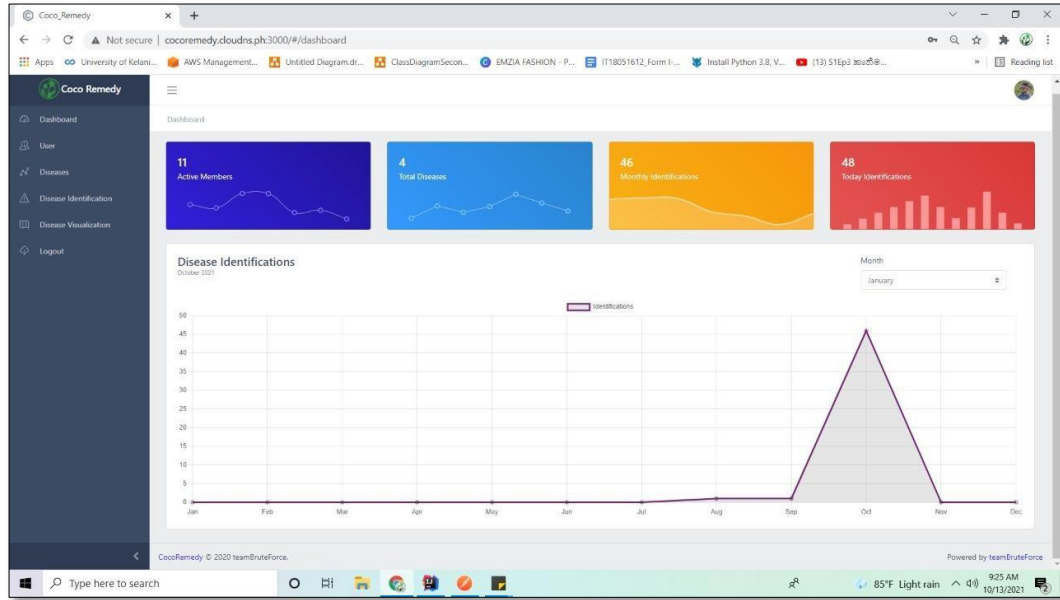


Figure 6. 13: Deployed web application



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](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](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](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](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](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% 	<a href="#">Resubmit</a> <a href="#">View</a> 
-------------------------------	---	---	---	---

### Appendix - B : Sample Questionnaire

<https://forms.gle/pXNQrAichCegcovi8>