

# **CROPGUARD: AI-DRIVEN IOT-BASED SMART FARMING SYSTEM**

**KE JOEY**

**UNIVERSITI TUNKU ABDUL RAHMAN**

# **CROPGUARD: AI-DRIVEN IOT-BASED SMART FARMING SYSTEM**

**KE JOEY**

**A project report submitted in partial fulfilment of the  
requirements for the award of  
Bachelor of Electronics Engineering with Honours**

**Faculty of Engineering and Green Technology  
Universiti Tunku Abdul Rahman**

**May 2025**

## DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : Jul-

Name : KE JOEY


ID No. : 20AGB04163

Date :

### APPROVAL FOR SUBMISSION

I certify that this project report entitled “**CROPGUARD: AI-DRIVEN IOT-BASED SMART FARMING SYSTEM**” was prepared by **KE JOEY** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Electronics Engineering with Honours at Universiti Tunku Abdul Rahman.

Approved by,

Signature : 

Supervisor : Ts. Dr. Toh Pek Lan

Date : 13 May 2025

Signature : \_\_\_\_\_

Co-Supervisor : \_\_\_\_\_

Date : \_\_\_\_\_

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2025, Ke Joey. All right reserved.

## **ACKNOWLEDGEMENTS**

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Dr. Toh Pek Lan for her invaluable advice, guidance and his enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parents and friends who had helped and given me encouragement throughout the time I spent on completing my final year project. Besides, I would like to thank Encik Khairul Hafiz Bin Mohamad for his insightful guidance on 3D printing.

## **CROPGUARD: AI-DRIVEN IOT-BASED SMART FARMING SYSTEM**

### **ABSTRACT**

Protecting crops against wildlife intrusions has become a pressing need as the delicate balance between human livelihoods and wildlife existence is disrupted. Reason behind this is that the encroachment of wildlife habitats caused by the rising socio-economic activities has led to human-wildlife conflict. Therefore, this scenario underscores the critical importance of developing an innovative crop protection system to mitigate the issue of crop raiding faced in agricultural sector. This project proposed an IoT-based smart farming system driven by Artificial Intelligence (AI). In other words, this system is developed in such a way that it builds on the foundation of IoT and AI to expand its system capabilities in protecting the crop against wildlife intrusions intelligently. This system is developed to have a variety of functionalities including detecting the presence of animal, classifying the animals and initiating appropriate actions to repel the threatening animals away. In this project, a Raspberry Pi 4 Model B acts as a central processing unit to process and interpret several data inputs from multiple sensors such as Raspberry Pi Camera Module 2, Passive Infrared (PIR) sensor and buzzer. Besides, machine learning model such as Haar Cascade Classifier is used to empower the system to detect and classify the animals into different categories. The collected data is processed and analysed in real-time. Real-time alert notification will be sent to the farmers upon wildlife intrusions. With this feature, the farmers can be informed immediately, enabling them to execute further actions to the animals and their crops.

## TABLE OF CONTENTS

<b>DECLARATION</b>	<b>ii</b>
<b>APPROVAL FOR SUBMISSION</b>	<b>iii</b>
<b>ACKNOWLEDGEMENTS</b>	<b>v</b>
<b>ABSTRACT</b>	<b>vi</b>
<b>TABLE OF CONTENTS</b>	<b>vii</b>
<b>LIST OF TABLES</b>	<b>x</b>
<b>LIST OF FIGURES</b>	<b>xi</b>
<b>LIST OF SYMBOLS / ABBREVIATIONS</b>	<b>xiv</b>
<b>LIST OF APPENDICES</b>	<b>xvi</b>

## CHAPTER

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Background	1
	1.2 Problem Statement	2
	1.3 Aims and Objectives	3
<b>2</b>	<b>LITERATURE REVIEW</b>	<b>4</b>
	2.1 Object Detection	4
	2.2 Machine Learning	5
	2.2.1 Few-shot Learning	5
	2.3 Deep Learning	7
	2.3.1 Deep Convolutional Neural Networks	9
	2.3.2 You Only Look Once	10
	2.4 Literature Review	12
	2.4.1 Smart Animal Repelling Device: Utilizing IoT and AI for Effective Anti-Adaptive	



	Harmful Animal Deterrence by Mishra and Yadav (2024)	12
	2.4.2 Edge AI in Sustainable Farming: Deep Learning-Driven IoT Framework to Safeguard Crops from Wildlife Threats by Reddy <i>et al.</i> (2024)	15
	2.4.3 A Review of Crop Protection Methods in Agricultural by Rubi <i>et al.</i> (2024)	16
	2.4.4 Airep: Ai and IoT Based Animal Recognition and Repelling System for Smart Farming by Lekhaa <i>et al.</i> (2022)	18
<b>3</b>	<b>METHODOLOGY</b>	<b>21</b>
	3.1 Design Architecture	21
	3.2 Hardware System Design	22
	3.2.1 Raspberry Pi 4 Model B	23
	3.2.2 Raspberry Pi Camera Module v2	24
	3.2.3 Passive Infrared Sensor	26
	3.2.4 Buzzer	27
	3.3 Software System Design	27
	3.3.1 Python Programming	28
	3.3.2 VNC Viewer	29
	3.3.3 Telegram	29
	3.4 Hard-Level Proposed System Design	30
	3.5 Project Management	31
<b>4</b>	<b>RESULTS AND DISCUSSIONS</b>	<b>34</b>
	4.1 Hardware Development	34
	4.1.1 Electronic Components Integration	34
	4.1.2 Enclosure Fabrication	35
	4.1.3 Prototype	38
	4.2 Software Development	40
	4.2.1 Software Setup	40
	4.2.2 Classification Model	41
	4.3 Unified System Workflow	44
	4.3.1 System-level Operation Workflow	45

4.3.2	Software Logic and Execution Flow	49
4.4	System Analysis and Evaluation	55
4.4.1	Detection Performance Analysis	55
4.4.2	Repelling Mechanism Analysis	60
4.4.3	Total Response Time Analysis	62
4.4.4	Condition-based Accuracy Analysis	63
<b>5</b>	<b>CONCLUSIONS AND RECOMMENDATIONS</b>	<b>69</b>
5.1	Conclusion	69
5.2	Recommendation	70
	<b>REFERENCES</b>	<b>71</b>
	<b>APPENDICES</b>	<b>74</b>

**LIST OF TABLES**

<b>TABLE</b>	<b>TITLE</b>	<b>PAGE</b>
3.1	The Specifications of Raspberry Pi 4 Model B (Raspberry Pi, 2019).	24
3.2	The Specifications of Raspberry Pi Camera Module v2 (Raspberry Pi, n.d.).	25
3.3	The Specifications of Passive Infrared (PIR) Sensor (Ada and Dicola, n.d.).	26
3.4	Gantt Chart for FYP 1.	32
3.5	Gantt Chart for FYP 2.	32

## LIST OF FIGURES

FIGURE	TITLE	PAGE
2.1	General Overview of Few-shot Learning Framework (Kundu, 2022).	7
2.2	Flowchart of Object Detection Framework (Sayagavi, Sudarshan, and Ravoor, 2021).	8
2.3	Overview of DCNN Architecture (Aloysius and Geetha, 2017).	10
2.4	The System Architecture of Smart Animal Repelling Device (Mishra and Yadav, 2024).	14
2.5	The System Architecture of the Proposed CNN (Lekhaa <i>et al.</i> , 2022).	19
3.1	Flowchart of the Proposed System.	22
3.2	Raspberry Pi 4 Model B (Raspberry Pi, 2019).	23
3.3	Raspberry Pi Camera Module v2 (Raspberry Pi, n.d.).	25
3.4	Passive Infrared Sensor (Global Sensor Technology, 2021).	26
3.5	Buzzer.	27
3.6	Logo of Python (Ra20Ga, 2021).	28
3.7	Logo of VNC Viewer (Uptodown, n.d).	29
3.8	Logo of Telegram (Logos-world, 2024).	30
3.9	Hard-Level Design of the Proposed System.	31
4.1	The Raspberry Pi's Desktop Interface through VNC Viewer.	35

4.2	3D Printer Home Setting for Raspberry Pi Camera Enclosure.	37
4.3	3D Printing for the Support Structure of the Prototype is in Progress.	37
4.4	Design Preview of Assembly Enclosure for Raspberry Pi Camera Module V2 and PIR Sensor in IdeaMaker Software.	38
4.5	Design Preview of Enclosure for Raspberry Pi 4 Model B in SolidWorks.	38
4.6	Prototype of the Smart Farming System.	40
4.7	Successful Installation of OpenCV library.	41
4.8	Working Concept of Integral Images for Object Detection (Mittal, 2020).	43
4.9	Part of the Code to Classify Cats based on Different Sets of Parameters.	43
4.10	Flowchart of Smart Farming System.	46
4.11	Images Saved in Folder.	47
4.12	Message in the Shell of System Program.	47
4.13	Alert Message in Telegram.	48
4.14	PuTTY Configuration.	51
4.15	Authentication of VNC Server.	51
4.16	Detection Log File for Event Tracking.	52
4.17	Input Frame.	52
4.18	Converted Grayscale Frame.	53
4.19	Window at Position (1,24).	53
4.20	Window at Position (60,60).	53
4.21	Window at Position (100,100).	53
4.22	Window at Position (130,50).	54
4.23	Window at Position (40,100).	54

4.24	Window at Position (30,150).	54
4.25	A cat is Successfully Detected.	54
4.26	Stabilization of PIR Sensor.	54
4.27	Confusion Matrix of the Classification Model.	57
4.28	Recognition and Classification Metrics of the System.	58
4.29	Detection Latency of Smart Farming System.	59
4.30	Repelling Latency of Smart Farming System.	61
4.31	Repelling Success Rate of Smart Farming System.	62
4.32	Total Response Time of Smart Farming System.	63
4.33	White Cat is Detected.	64
4.34	Gray Cat is Detected.	65
4.35	Orange Cat is Detected.	65
4.36	Cat is Detected in Dark Condition.	66
4.37	Short Detection Distance.	67
4.38	Long Detection Distance.	67
4.39	Multiple Cats are Detected.	68

## LIST OF SYMBOLS / ABBREVIATIONS

fps	Frames Per Second
AI	Artificial Intelligence
API	Application Programming Interface
CNNs	Convolutional Neural Networks
CPU	Central Processing Unit
CSRT	Channel and Spatial Reliability Tracking
DCNN	Deep Convolutional Neural Networks
DPM	Deformable Parts Model
FN	False Negative
FP	False Positive
GPIO	General-Purpose Input and Output
GSM	Global System for Mobile communication
HOG	Histogram of Oriented Gradients
HTTP	HyperText Transfer Protocol
HWC	Human-Wildlife Conflict
IOU	Intersection Over Union
IOS	Iphone Operating System
IoT	Internet of Things
LoRa	Low Range
NMS	Non-Max Suppression
OS	Operating System
PIR	Passive Infrared
RAM	Random Access Memory
RFID	Radio Frequency Identification
R-CNN	Recursive Convolutional Neural Network
R-CNN	Region-based Convolutional Neural Network

SARD	Smart Animal Repelling Device
SIFT	Scale-Invariant Feature Transform
SSH	Secure Shell
SSMD	Single Shot Multibox Detector
SVM	Support Vector Machines
TN	True Negative
TP	True Positive
UAVs	Unmanned Aerial Vehicles
VNC	Virtual Network Computing
W-Co-Hog	Weighted Co-occurrence Histograms of Oriented Gradients
XML	eXtensible Markup Language
YOLO	You Only Look Once
3D	3-Dimensional



**LIST OF APPENDICES**

<b>APPENDIX</b>	<b>TITLE</b>	<b>PAGE</b>
A	Python Program for Integrated Smart Farming System.	74
B	Program Code to Create Vector File of Positive Samples for Training Automation in Bash.	79
C	Program Code to Generate Annotated Training Data for the Classification Model.	81

## CHAPTER 1

### INTRODUCTION

#### 1.1 Background

As the world population is growing, the demand for food production also increases over the time. At the same time, the rapid growth in food consumption also leads to the increase in food demand. For this reason, a sustainable global food system is crucial to maintain the balance between food supply and demand, ensuring food security. Unfortunately, the food production is often interrupted by the conflict arises between human and wildlife. The wild animals invade into the crop field and cause a significant damage to the crop. According to the research, there is a serious loss of household food requirements which caused by the wild animals' intrusions occurred in two districts of Bhutan: Tsirang and Trashiyangste (Wangchuk *et al.*, 2023). To make the matter worse, there has been a noticeable decline in farming since the food production is greatly hindered by the wild animals. As a result, the workforce in agricultural sector also decreases dramatically. Human-Wildlife Conflict (HWC) remains a pertinent and ongoing challenge for farmers in agricultural sector especially in rural countries. Crop damage brings by HWC not only leads to food shortages but also family migration. This is because the farming families have lost their interest in farming, thus abandoning their farmland and migrate to other countries for living. The implication of this issue extends beyond the disruption of food production, resulting in substantial financial loss especially to those farmers who mainly depend on the crops for their daily income. Therefore, this scenario underscores the critical importance of developing an innovative smart farming system in agricultural sector. As its name

suggests, a smart farming system adopts the concept of Internet of Things (IoT) to address this issue faced in food production system. With these features incorporated in a smart farming system, farmers are able to mitigate the impact of wildlife invasions to their crop field. Besides, it reduces the necessity to employ a large number of labour workforce to monitor the crops. A smart farming system utilizes integration of IoT with artificial intelligence (AI) to repel the wildlife away from the crop field in an intelligent way before the wildlife manage to destroy the crops. The smart farming system is designed in such a way that it is able to detect the animals, classify the animals, capture the image of the animals, repel the animals and send alert message to the farmers via mobile applications.

## **1.2 Problem Statement**

For many years, several approaches have been implemented to address the problem of crop raiding due to the wildlife invasions (Rubi *et al.*, 2024). Approaches such as chemical repellent or releasing smoke is no longer effective to repel the animals away as the animals already have adaptability to these approaches as time goes by. Besides, traditional animal repelling methods including physical barriers such as electric fences could be harmful to the animals and even lead the animals to death. While these methods offer promising results, but there are many major drawbacks exist in these traditional methods. For this reason, progression from traditional methods to more advanced methods should be emphasized to expand the system's capability in protecting the crops from wildlife. The proposed system in this project is going to develop an AI-driven IoT-based smart farming system. This system is designed to repel the animals in an intelligent, efficient and autonomous way. Unlike the traditional animal repelling methods, this system is capable to repel the animals accurately and consistently without causing physical damage to the animals. Hence, unnecessary harm to the animals could be avoided. Apart from that, this proposed system is able to send alert notification to the farmers upon animal invasions. The farmers will be able to monitor the crops condition and execute further actions to the intruding animals.

Hence, this proposed system is a leap forward in crop protection as it integrates advanced technologies and artificial intelligence to offer a smart farming system.

### **1.3 Aims and Objectives**

The objectives of this project are shown as follow:

- i. To develop a system which can detect the presence of the wild animals, classify the animals, and deter the animals away from the crop field.
- ii. To develop a system which can interact with IoT cloud to monitor the crop condition and send a real-time alert message to the farmers.
- iii. To develop a software program using machine learning algorithms to classify the animals detected by the system.

## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1 Object Detection**

Object detection is a technique of detecting the object by locating the object and classifying the objects within the image. In simpler words, identification of what is the object and where is the object in the image is the key component of objection detection. The object is first localized by drawing a bounding box around it to denote its location. If there are multiple objects present within the images, then there will be multiple bounding boxes to locate the objects within the images. When it comes to object localization, the coordinate of the bounding box plays its important role to determine the location of the object. The image will lie below the x-axis and at the right of y-axis. Therefore, the coordinate of the object within the image will be calculated towards a downward direction. At the same time, process of object classification takes place where the objects are classified into each of their respective categories. For instance, trees, flowers, chairs, and others. This classification is carried out based on the extracted features from the object. According to Kellenberger, Volpi, and Tuia (2017), the traditional way of object detection in computer vision adopts the technique of hand-crafted features. For example, Scale-Invariant Feature Transform (SIFT) and Histogram of Oriented Gradients (HOG). At its essence, these techniques are considered impractical as both methods require intensive computations which in turns make these methods a major roadblock to the real time applications. Besides, manual annotations that involved in traditional object detection is time consuming and a lot of professional experts are required to perform this task. That is when Convolutional

Neutral Networks (CNNs) come into rescue to introduce a more advanced and effective system to perform the object detection in computer vision. CNNs offers end-to-end, task-specific joint feature and classification learning (Kellenberger, Volpi, and Tuia, 2017). Besides, the researchers have suggested a CNN architecture, specifically optimized to detect small and similarly sized objects in Unmanned Aerial Vehicles (UAVs) images in a faster and more effective way.

## **2.2 Machine Learning**

Machine learning is considered as a technique that breaks down from artificial intelligence (AI) to perform learning and decision-making from the given data. It is a cutting-edge approach in recognizing image, automating advanced tasks and so on through the process of self-learning. In general, a conventional object detection often requires a large number of training samples for Artificial Intelligence (AI) models to recognize different classes of data. In other words, several hundreds or even thousands of labelled data points are required to train these deep learning AI models for a better accuracy in the data recognition. However, it is a challenging approach to obtain a large number of samples in real-world scenarios. The reasons are chiefly as follows: there are many rare and newly discovered species which leads to the lack of samples available. Besides, since data annotation involves domain-specific expertise, thus obtaining labelled samples can be a high expenses activity. For this reason, the performance of conventional object detection methods is not desirable. In this case, few-shot learning is introduced to solve this issue.

### **2.2.1 Few-shot Learning**

Few-shot learning is a machine learning framework that allows the AI models to provide accurate predictions with limited number of data samples (Thailappan, 2024). This implies that few-shot object detection can perform better than the conventional

object detection under the case of few data training samples. According to Feng and Xiao (2022), few-shot learning serves its purpose to develop the ability in learning and autonomous generalization from a limited number of data samples. When it comes to few-shot learning, generalization has more emphasis over memorization. It contributes to rapid generalization to new task with supervised information on a limited number of data samples. In applications, few-shot learning is widely used in the field of computer vision especially object detection, image recognition and classification. The working principle of few-shot learning involves N-way K-shot classification where N refers to the number of classes while K refers to the number of samples provided for each class. Figure 2.1 illustrates a general overview of a few-shot learning framework. Based on the figure, the framework consists of support set and query set (Kundu, 2022). The support set is considered as a 3-way 3-shot classification since the support set consists of three different classes (penguin, pelican and puffin) and each class has three different examples which arranged in a vertical way. Basically, the support set helps the pre-trained models to learn how to generalize representations for each class. Other than that, the query set provides new examples for each class after prediction of classification has been done by the model based on the generalized representations learned from the previous support set. As a result, there are one new example provided for each respective class such as penguin, pelican and puffin. From what has been discussed above, it can be recognized that a lower value of N is more preferable to ensure the accuracy of few-shot learning. This is because lower N-way indicates that the pre-trained model needs to generalize over a smaller number of classes for representations. In simpler words, accuracy in prediction of classification increases when the number of classes decreases. On the contrary, prediction accuracy increases when K, number of samples provided for each class increases. This is because higher value of K indicates that more samples are given to support the models in making accurate similarity predictions. Apart from this, training a function to predict similarity between the classes in the support set and query set is the fundamental concept of few-shot learning. For instance, given that the image provided in the query set is a penguin, it is being compared to the classes such as penguin, pelican and puffin in the support set. By comparing the query image with penguin, the outcome of similarity function is 0.9. However, when the query image is being compared with pelican, the outcome of the similarity function will be 0.2. In the case of comparison with puffin, the similarity

score is 0.4. Therefore, the class with the highest similarity score will be filtered and placed at the top of the query set predictions as shown in Figure 2.1.

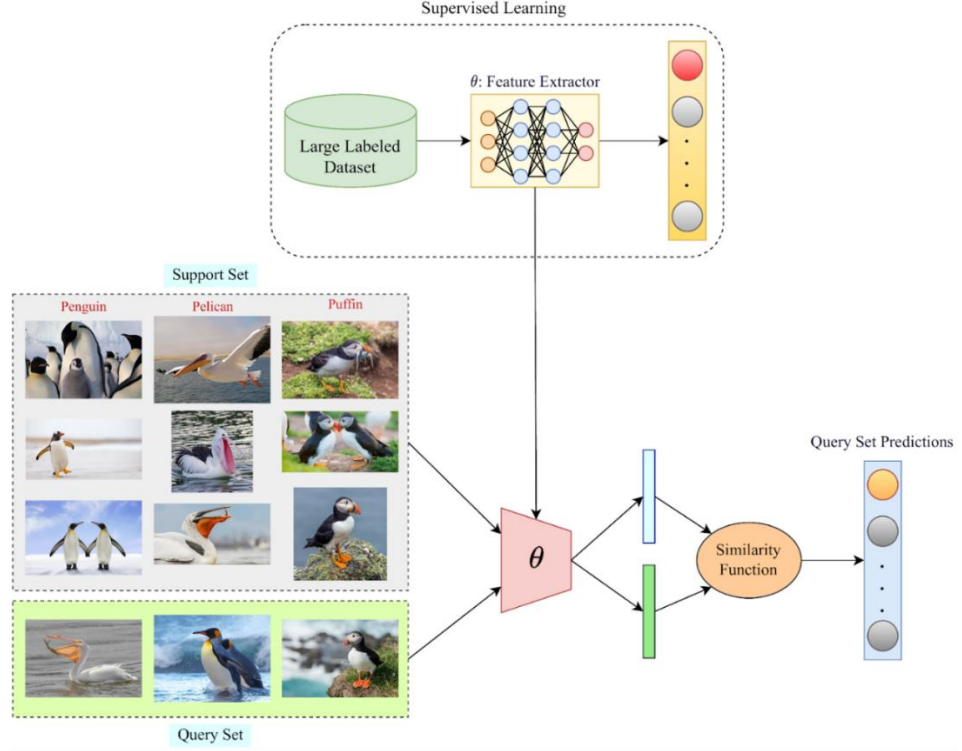


Figure 2.1: General Overview of Few-shot Learning Framework (Kundu, 2022).

### 2.3 Deep Learning

With the purpose of developing an energy-efficient system, Sayagavi, Sudarshan, and Ravoor (2021) have proposed an intelligent system that will only activate the camera to capture the image in response to movement detection. Besides, this proposed system also introduces an advanced technique to ascertain the motives of the animal detected and generate appropriate action to the animals in response with the motives analysed. This will be a significant approach to minimize false positives which can lead to unnecessary damage to the animals and wasted resources. A case in point, there are some birds that intend to prey on the rodents which feed on crop in the crop field. In this case, the birds do not pose any threat to the crop. However, once the system detects the presence of the birds without ascertaining their motives, it automatically releases a



strong chemical deterrent to drive the birds that actually help with pest control. Consequently, the ultimate goal of the system in protecting the crop will not be achieved. Up to a point, the local ecosystem can even be disrupted by the immature system developed to protect the crop. According to the research carried out by Sayagavi, Sudarshan, and Ravoor (2021), You Only Look Once (YOLO) model is chosen to be the object detection method due to its outstanding performance to provide fast and accurate result. Figure 2.2 provides the flowchart of object detection framework introduced by YOLO model. The process of Non-Max Suppression (NMS) serves its purpose to eliminate low confidence and duplicate detections. According to the flowchart, object tracking takes place if there is wild animal being detected in the frame. In this case, Channel and Spatial Reliability Tracking (CSRT) tracker is used as the object tracker to ascertain the motive of the animals detected in the frame. Furthermore, a set of rules are predefined and used as a reference to execute appropriate actions. This is because if the animal's motive aligns with the predefined rules, appropriate actions will be taken in order to protect the crop. In terms of hardware design, Raspberry Pi is implemented for the purpose of video capture and video processing. Two Raspberry Pi are required for this proposed system to balance

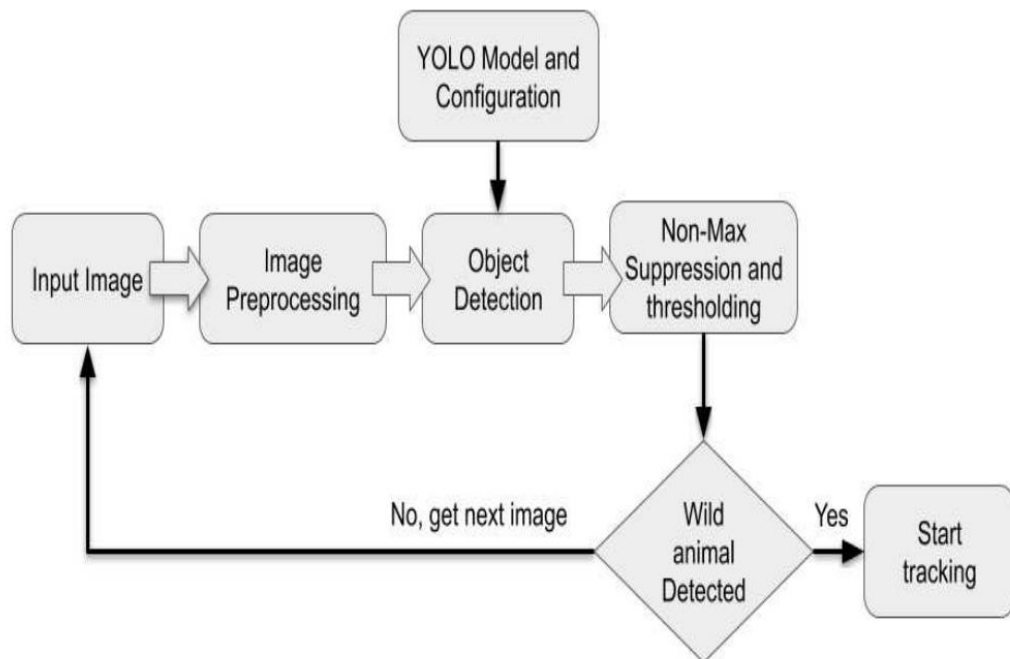


Figure 2.2: Flowchart of Object Detection Framework (Sayagavi, Sudarshan, and Ravoor, 2021).

the processing load. Besides, the process of inter-device communication also requires two Raspberry Pi to communicate with each other about the tracking condition. By way of example, if the object is out of sight of the camera of the first Raspberry Pi, a message will be sent to the second Raspberry Pi to indicate loss track of the object. This process is known as hand-off.

### 2.3.1 Deep Convolutional Neural Networks

As its name implies, Deep Convolutional Neural Networks (DCNN) is Convolutional Neural Networks (CNN) that involves a greater number of layers. This is the reason it is being called “deep” since it composed of many convolutional layers which leads to a greater depth. Figure 2.3 illustrates the overview of a DCNN architecture (Aloysius and Geetha, 2017). By referring to the figure, it can be seen clearly that DCNN has different types of layers such as convolutional layer, pooling layer and fully connected layers. In the first place, the convolutional layer will take an image as its input. The convolutional layer is made up of several feature maps which consist of neurons that are responsive to a particular region of the input image. During this stage, multiple computations will take place and convolution operations are being performed. Two-dimensional (2D) activation maps are produced from the combination of filters and feature maps by convolution. Since several filters are used to convolve with the feature maps thus the filters overlap with each other in the convolutional layer as shown in Figure 2.3. The output volume will then be generated from the stacking of activation maps along the depth dimension. In simpler words, the filters which are also called as kernels will slide over the input image several times. This process is known as convolution operations. As a result, activation maps that emphasize the extracted feature of the input image will be produced. Next, DCNN proceeds to the following stage which involves pooling layer. During this stage, it can be observed obviously that the feature maps which are also known as activation layers have reduced their size from  $20 \times 20$  to  $10 \times 10$ . This is because pooling layer functions to decrease the size of the activation maps while retaining its original information. Consequently, the number of computations and parameters will also decrease which leads to

minimization of overfitting effect. The convolution process and pooling process are then repeated to boost the effectiveness of features and pattern extraction from the images and enhance object detection and classification. DCNN then proceeds to the next stage includes fully connected layer after the final pooling layer. As its name suggests, fully connected layer implies that in this layer, the neurons in convolutional layer, pooling layer and this layer are fully connected to each other. Fully connected layer is the final layer in DCNN architecture before the final classification. Therefore, this layer plays an important role to perform high-level reasoning. Based on the DCNN architecture, fully connected layers evolve from  $4 \times 4$  to  $1 \times 1$ . This is due to the reason that fully connected layers receive an input vector from the previous layer which has been flattened into one dimensional vector. In other words, the image pixels of the output from previous layers are flattened and treated as the input of fully connected layer. Finally, fully connected layers perform high-level reasoning which includes application of Softmax function that leads to classification of image. As a result, classifier is produced from fully connected layer.

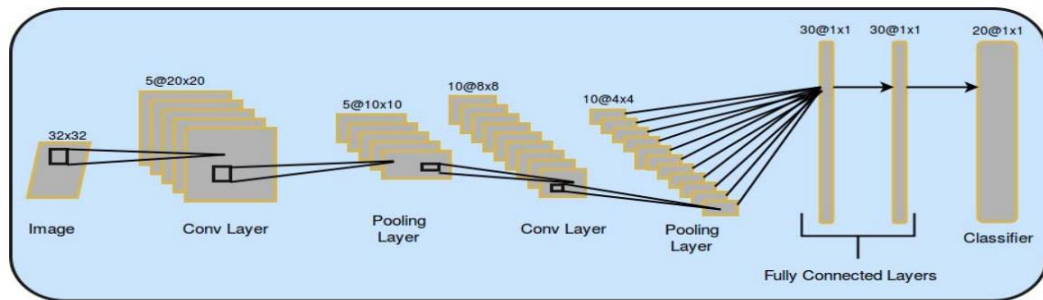


Figure 2.3: Overview of DCNN Architecture (Aloysius and Geetha, 2017).

### 2.3.2 You Only Look Once

As its name implies, You Only Look Once (YOLO) is a real-time object detection algorithm which the YOLO model can detect and classify the object within an image by only looking once at that image. Redmon *et al.* (2016) indicated that YOLO is an effective approach in object detection, which performs better than other object detectors, such as Region-based Convolutional Neural Network (R-CNN) and

Deformable Parts Model (DPM). The reason behind this is that R-CNN is a region proposal-based models where it involves multi-stage process, and its pipelines are complex. Thus, R-CNN often requires more time in processing the region proposals hence it is slow during inference. In addition, DPM adopts the concept of sliding window-based which the classifier performs object recognition at regular intervals across the entire image. As a result, the overall process will be slowed down. Due to the common practical limitation of slow processing introduced by other object detectors, YOLO stands out for its capability to provide fast and accurate real-time object detection. YOLO detects and classifies the object simultaneously by looking once at an image. Meanwhile, its detection is framed as a regression problem therefore the complex pipelines such as region proposals and classifiers are abandoned. For this reason, YOLO can perform image processing at 45 frames per second (fps). To certain extent, a fast YOLO network can upsurge to run at 155 fps (Redmon *et al.*, 2016). YOLO emphasized on real-time speed and accuracy since it can be used to process real-time image and streaming video with low level of latency. In addition, YOLO serves a purpose to boost the generalization for new domains (Keita, 2022). This implies that YOLO can generalize on different shapes of objects, hence it is an effective approach when it comes to applications that need rapid and robust object detection. Besides that, YOLO works on global reasoning where it observes the full image during both stages including training and testing. In this case, both contextual information and appearance information can be encoded implicitly. By doing so, mistakes such as false positives prediction on background can be greatly avoided as YOLO can view a greater context and even perform better prediction in situations where the objects seem ambiguous or presents in various contexts. Despite the fact that YOLO provide better object detection performance than other object detection methods, it still comes with its limitation. Localization of small objects that come in groups such as swarm of fishes is an issue for YOLO due to the spatial constraints introduced by YOLO. YOLO employs the technique of unified detection which involves integration of various object detection components into a single neural network. Generally, YOLO algorithm work on the basis of grid division, bounding box regressions, intersection over union (IOU), class probabilities prediction, and non-maximum suppression (NMS) (Kundu, 2023). In terms of applications, YOLO is an effective object detector in the field of agriculture, healthcare, self-driving cars and

security surveillance (Keita, 2022). For instance, YOLO is implemented in vision-based robots such as harvesting robots to identify and harvest the target fruits and vegetables accurately and effectively.

## **2.4 Literature Review**

In this study, several literatures are reviewed to develop a better understanding of the existing research and technologies that are relevant to this area of study. By conducting literature reviews concerning crop protection, a basic conceptual framework of smart farming system is identified and incorporated in the development of this project. The strengths and weaknesses of each literature review are discussed in this chapter.

### **2.4.1 Smart Animal Repelling Device: Utilizing IoT and AI for Effective Anti-Adaptive Harmful Animal Deterrence by Mishra and Yadav (2024)**

This study focuses on the study of Smart Animal Repelling Device (SARD) by incorporating the elements of Internet of Things (IoT) and Artificial Intelligence (AI). Mishra and Yadav (2024) introduced a SARD framework which runs IoT applications in microservices by abiding the principle of containerization and uses Docker containers. According to the research work, it is a necessity to deploy a smart and flexible deterrent method to replace the traditional methods. The reason behind this is that the traditional deterrent methods such as physical barriers and fear tactics are no longer practical and effective in repelling the animals away as the animals have great adaptability toward these traditional methods as time goes by. With the integration of IoT and AI in animal repelling methods, the crop losses have been significantly reduced to 30% which reflects the effectiveness of SARD using IoT and AI. There are several reasons for the effectiveness of SARD, but they come down to three major contributions. Firstly, SARD uses the technique of combination of various components such as a Passive Infrared (PIR) sensor, solar panel, and Low Range (LoRa)

technology to ensure that the real-time animal detection is energy saving (Mishra and Yadav, 2024). Detection of the infrared radiation emitted by objects within the field of view of PIR sensor is the function of a PIR sensor. By way of explanation, the infrared energy pattern will change once an object motion is detected within the field of PIR sensor's view. PIR sensor will capture the changes in the infrared energy pattern, and it will be triggered by the changes to send a signal to a linked responsive device to initiate an appropriate action. In the context of designing a faster and more accurate real-time animal identification system, Single Shot Multibox Detector (SSMD) is used in conjunction with the Recursive Convolutional Neural Network (R-CNN) model on edge devices. This is because SSMD is an enhanced version of previous single shot detector (YOLO), and it is much more accurate and precise with slower techniques that carry out explicit region proposals and pooling such as Faster R-CNN. Besides, the key to the accuracy of SSMD lies in the elimination of the process of resampling feature and pixels for bounding box proposals that is performed by SSMD. Hence, there is no doubt that this approach contributes significantly to the effectiveness of SARD in terms of speed and accuracy. The third contribution comes from the re-identification designs. With the re-identification designs, the effectiveness of an animal deterrent can be assured by the system architecture through identity association, monitoring and real-time alerts. In other words, when re-identification is incorporated into the system architecture of SARD, SARD can identify and monitor the individual animals over time. Likewise, SARD contributes to monitoring the animal's behaviour and initiate action with real-time alerts. For this reason, it allows SARD to perform effectively to repel the animals away. Besides, the researchers also highlight the intelligent devices used in SARD which cover identification and ultrasound production, real-time detection and integration among animal identification and repelling device. All in all, SARD serves its purpose to provide real-time object detection by using various comprehensive approaches such as PIR sensors, ultrasonic emission and edge computing. Figure 2.4 illustrates the SARD system architecture. It can be observed clearly that the overall system architecture comprised of IoT, AI, and edge computing. The integration of IoT, AI and edge computing facilitates the level of intelligence of a smart animal repelling device. Despite the fact that the SARD system has proven to be a significant tool in the aspect of animal detection and repelling action, but there are still rooms for improvement. Ultrasonic frequencies remain as an issue in this study.

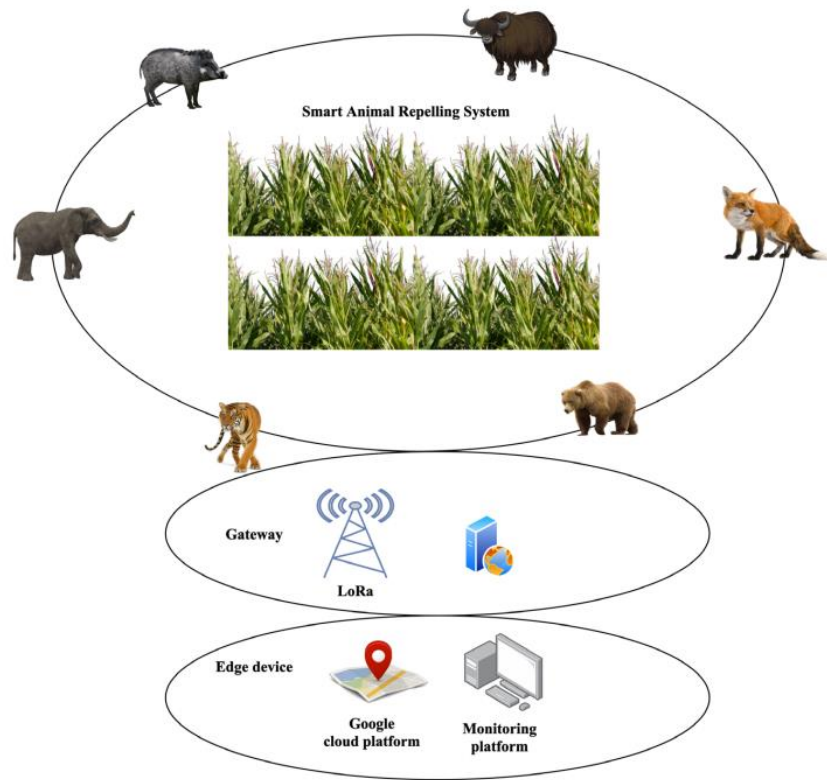


Figure 2.4: The System Architecture of Smart Animal Repelling Device (Mishra and Yadav, 2024).

Although SARD utilizes variety of ultrasound frequencies which tailored to the specified animals based on their classification, but there is still a need to refine the ultrasonic frequency in terms of its functionality and frequency modulation. Another limitation of this study is the power efficiency of the SARD system which can be further improved. There are several proposed solutions recommended to address the issues in this study. More advanced algorithms involving machine learning and artificial intelligence models are taken into consideration to investigate the effectiveness of ultrasonic frequencies that caters to different animals. Other than that, precise and careful calibration of ultrasonic emission should be implemented to boost the effectiveness of ultrasonic emission across a wide spectrum. Since the system is designed to function for 24 hours to keep track the conditions of crop field in a remote area, therefore power efficiency of the system should be maximized. In this case, the proposed solution is introducing solar panel system to replace the energy resources such as batteries. Since this system will be placed in outdoor areas, therefore natural renewable resources should be utilized to enhance the power efficiency of this system.

#### **2.4.2 Edge AI in Sustainable Farming: Deep Learning-Driven IoT Framework to Safeguard Crops from Wildlife Threats by Reddy *et al.* (2024)**

This research work highlights the pressing needs to incorporate artificial intelligence in sustainable agriculture. Research has shown that the crop production in agriculture is constantly hampered by animal intrusion which eventually leads to crop damage. To certain extent, food production is being hindered by the crop damage that arises. Meanwhile, the imbalance of supply and demand also become worsen. Reddy *et al.* (2024) make a binding effort to propose a cutting-edge approach that integrates Internet of Things (IoT) with tinyML-based deep learning algorithms to leverage the potential of Edge AI in developing a sustainable wildlife intrusion detection and deterrence system. In this case, the system is developed in such a way that it synthesized an AI-CAM with a laser detection system for the purpose of ensuring an effective wildlife intrusion detection and classification. On the other hand, Internet of Things (IoT) is a significant technique that is used to allow the farmers to have the real-time access to the situation of the crop field. This research paper has suggested a lightweight deep learning model known as EvoNet to classify the animals. All in all, the proposed model revolves around the integration of IoT with AI. Apart from protecting crops from the wildlife, the proposed model is also designed to support ecological conservation while safeguarding the lives of agricultural communities. When it comes to wildlife detection system, Passive Infrared (PIR) sensor is commonly used to detect animal's motion. Nevertheless, PIR sensor has its drawbacks such as it is more likely to malfunction when its surrounding temperature exceeds 35°. This problem makes the wildlife detection system using PIR sensor become impractical in most of the countries that normally have hotter weather with temperature above 35°. Due to several practical limitations of PIR sensor, the researchers have come up with an innovative solution to address these problems. Laser-based boundary system is presented and utilized for wildlife intrusion detection system. The laser-based boundary system works in such a way that it produces laser which covers the boundary edge around the crop field. In other words, the laser emitted by the laser diode equipped in the laser-based boundary system will form a security barrier around the crop field. If there is any potential wildlife invasion, the laser boundary will be disrupted thus presence of wildlife will be detected in an early stage. Conversely, if



there is no wildlife invasion, the system is able to detect the absence of the wildlife and minimize its power consumption. At the same time, AI-CAM that is incorporated in the system will be responsible to capture the images of intruding animal once wildlife intrusion is detected. Consequently, the captured images will be analysed by the deep learning model to proceed with animal classification. In this case, TensorFlow Lite is used to implement the deep learning model. After animal classification is done, the central pole which is part of the system will carry out appropriate actions to deter the intruding animals away from the crop field. With the implementation of IoT in the system, the microcontroller embedded in the central pole is allowed to have Wi-Fi connectivity. Therefore, the central pole can send alert message to the farmer immediately once wildlife intrusion is detected and confirmed. In addition, the farmer can monitor the condition of the crop to double check whether the intruding animal has been deterred away from the crop field or not. This can be done by the wireless controller synthesized in a mobile robotic device known as rover. The rover has a camera that is placed at the front of the rover, it allows the farmers to have real-time access of their crop. In simpler words, they can monitor their crop field with live video stream provided in their mobile devices. Therefore, the farmers can now deploy any further strong deterrence action towards the intruding animal that still remain in the crop field by using the wireless controller in his hand. This instrumental approach offers a great deal of convenience to those farmers who are outstation. The limitation from this study is the limited computational power offered by TinyML. In this case, the suggestion is to develop more efficient algorithms to reduce the computational burden in resource constraint environment.

#### **2.4.3 A Review of Crop Protection Methods in Agricultural by Rubi *et al.* (2024)**

In this study, Rubi *et al.* (2024) summarised the limitations offered by various crop protection methods. Firstly, traditional crop protection methods such as electric and wire fences require a large amount of cost for installation. Besides, electric fences often exert negative impacts on wildlife by causing damage to the intruding wildlife.

Meanwhile, regular maintenance needs to be invested upon electric fence installation to avoid ineffective electric fences due to the vegetation that grow around the electric fence lines. Furthermore, Convolutional Neural Network (CNN)-based animal detection system needs a great deal of upfront efforts including significant setup and training time, and costly technology implementation. In other words, CNN-based animal detection system involves great amount of labelled training data and computation, thus the training time tends to be much longer. The same goes for laser fence as laser fence also require high expenses to implement its laser technology. Likewise, initial setup and technology adoption also remains an ongoing challenge for laser fences. Another crop protection method which is known as Weighted Co-occurrence Histograms of Oriented Gradients (W-Co-HOG) algorithm also face the same challenges as the previous crop protection methods that have been discussed before. For instance, it needs great time investment for initial setup and implementation, training and testing time. Based on the findings obtained in the study, there is no denying fact that the adoption of initial setup and technology is a common issue for most of the crop protection methods. For instance, laser fence, Internet of Things (IoT)-based monitoring system and smart agriculture with ultrasound emission. In addition, another crop protection method which is acoustic system with Global System for Mobile communication (GSM) is used to deter intruding animals by emitting sounds that are tailored to the fear of that animal. However, its sound effectiveness is constrained by certain range, and it relies on GSM networks. This implies that this acoustic system with GSM is more likely to lose its function in agricultural field that has a broader area. At the same time, this system might not be applicable to those rural areas that struggle to have a good network coverage. As a matter of fact, wildlife invasions often take place in those rural areas. From what has been discussed above, it is necessary to develop a structured approach to address the issues face by the current crop protection system. You Only Look Once (YOLO) version 5 is the enhanced YOLO model proposed by Rubi *et al.* (2024) to solve the problems of the current crop protection system. In this case, YOLOv5 is used in object detection, features extraction and object classification. However, YOLOv5 comes with its limitations. For instance, it is hard to detect small objects. Therefore, super-resolution techniques are suggested to improve input image preprocessing should be

implemented to address this issue. Besides, image reconstruction algorithms can be implemented to extract features of small objects.

#### **2.4.4 Airep: Ai and IoT Based Animal Recognition and Repelling System for Smart Farming by Lekhaa *et al.* (2022)**

In this work, Lekhaa *et al.* (2022) discussed the challenges of the traditional approaches and propose a system solution related to Artificial Intelligence (AI) Computer Vision based Deep Convolutional Neural Network (DCNN) for animal detection and ultrasonic emission used to repel the animals. According to the findings of their study, traditional approaches such as chemical repellents, electric fences, smoke, and others have exerted negative impacts on the surrounding environment of the monitored area. Chemical repellent or smoke causes environmental pollution leading to the disruption of ecological conservation. To a certain extent, strong chemical repellent or electric fences could pose a formidable threat to human and animals which is not the purpose of the smart farming system. Therefore, advancement of technology give rise to an intelligent sensor-based animal intrusion detection system. In other words, multiple sensors are incorporated into the system for motion detection and capture the image upon camera activation. After that, the system will proceed to the next stage which is image processing and classification using several techniques of predefined algorithms such as Weighted Co-occurrence Histograms of Oriented Gradients (W-Co HoG) feature vector. Furthermore, Support Vector Machines (SVM) is a machine learning algorithm used in the application of animal classification. When it comes to the sending of alert messages to the farmers, methods such as GSM module and Radio Frequency Identification (RFID) are responsible to enable the communication to the farmers. However, there are several disadvantages in the existing animal intrusion system. The common challenge is sensor failure since the existing animal intrusion system utilizes a lot of sensors. Therefore, the system proposed by Lekhaa *et al.* (2022) deploy DCNN algorithms and integrate AI-based computer vision methods with IoT devices to develop a holistic approach in protecting the crops. Deep Convolutional Neural Networks (DCNN) is the deep learning model

which is known for its effectiveness in performing tasks involving image recognition and image classification. In addition, the crop protection system functions to emit ultrasound which creates disturbance to the animals and prompts the animals to leave the crop field. Meanwhile, a notification system is set up to alert the farmers about the animal invasion and the condition of the crop field. In the context of designing a crop protection system, Lekhaa *et al.* (2022) have come up with a modules list which includes animal repellent web dashboard, animal recognition, repellent, monitoring and visualizing, notification and performance analysis. In the animal recognition module, training and test data annotation is required. Besides, pre-processing, animal detection, feature extraction, animal classification and prediction are the steps included in animal recognition module. Figure 2.5 shows the design architecture of the crop protection system proposed in this study. In conclusion, the proposed CNN developed

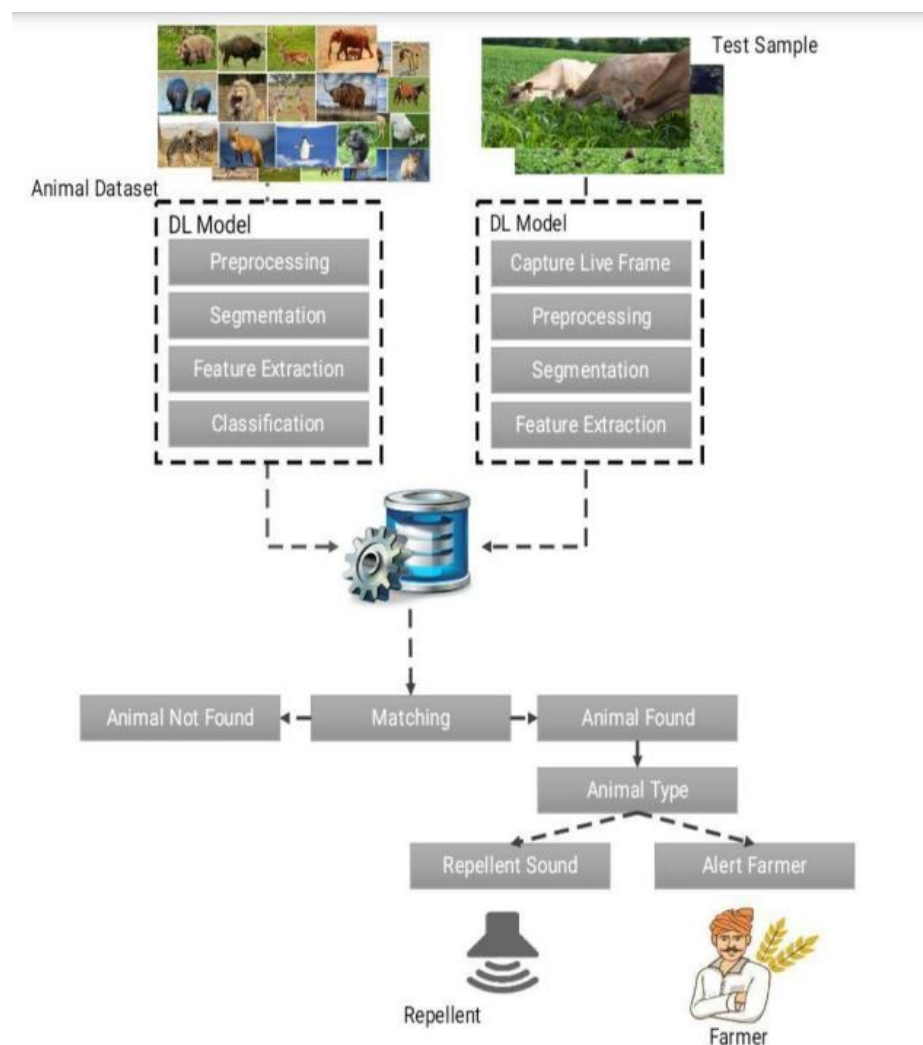


Figure 2.5: The System Architecture of the Proposed CNN (Lekhaa *et al.*, 2022).

by the researchers has shown a positive result where the accuracy in recognizing animals is about 98% which is relatively high. Nevertheless, there is still possibility of facing inaccurate results in the proposed system which is considered as the limitation of this study. For instance, false positives which the system detect presence of animal but in fact the animal is absent and false negatives which the system fail to detect the presence of animal where it supposed to. In order to address this issue, training and test data annotation needs to be enhanced by collecting more data for the pre-training model. As a result, a larger dataset will lead the deep learning model and machine learning model to obtain a better generalization with lesser mistakes.

## **CHAPTER 3**

### **METHODOLOGY**

#### **3.1 Design Architecture**

The design architecture of the AI-driven IoT-based Smart Farming System is illustrated as a block diagram as shown in Figure 3.1. In the beginning, an environment of the proposed system must be set up including the integration of hardware design with software design. In this context, the hardware equipments needed are Raspberry Pi 4 Model B, Raspberry Pi Camera Module v2, Passive Infrared (PIR) Sensor and a buzzer. As a result, a prototype for the proposed system will be produced. The hardware design of the proposed system is responsible for the motion detection, image capture and repellent action. If motion is detected by PIR sensor, the Raspberry Pi Camera Module v2 will be activated to switch on and observe the surrounding environment. The live environment will be monitored by the Pi camera if motion is detected. It is treated as the input frame of the software model in the following steps. The input frame is then pre-processed to enhance the image for object detection in computer vision. Pre-processing involves normalization, image resizing, noise reduction, and so on. After that, the image data is being fed into a machine learning model to classify the class of the image captured. Machine learning model such as Haar Cascade classifier is used for image classification. The image is classified into different categories. Since this proposed system targets animals like cat, hence object matching is required to determine whether the image captured matches the targeted animals. Once targeted animal is detected, the proposed system carries on with the next stage which involves image acquisition. Simultaneously, actions such as buzzing

sound emission and real-time alert notification system will be activated at the same time. In this context, hardware and software components are effectively well-integrated to demonstrate seamless operation in protecting the crop field.

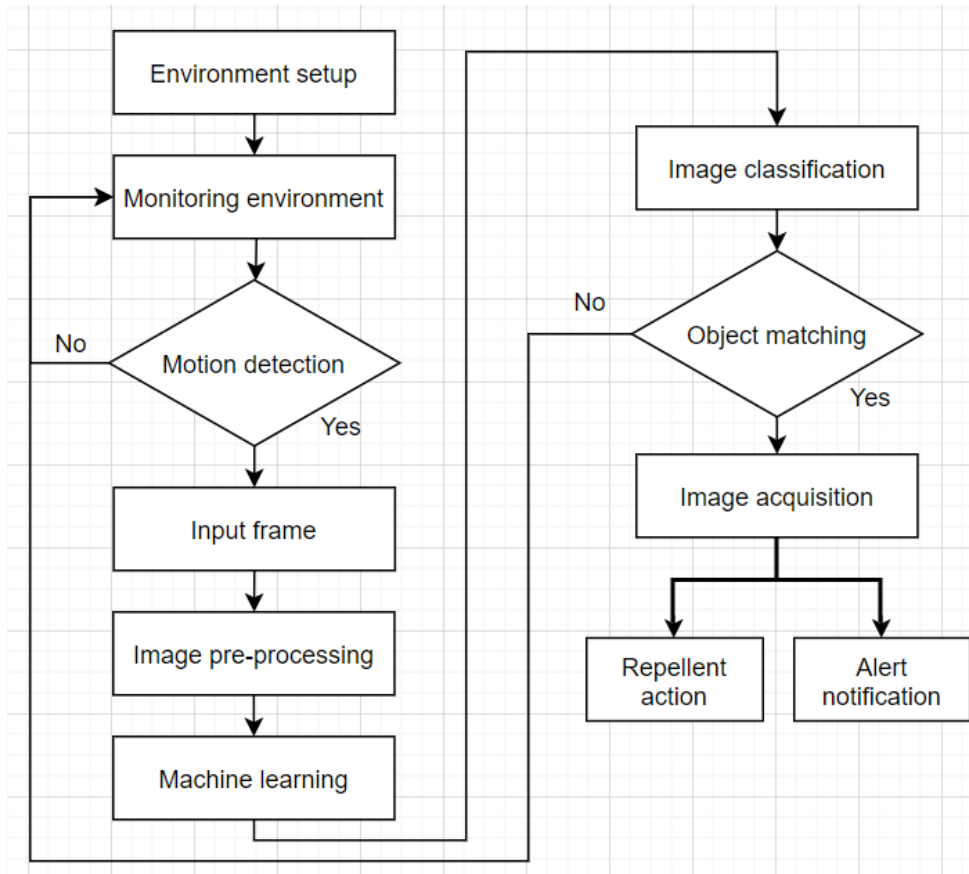


Figure 3.1: Flowchart of the Proposed System.

### 3.2 Hardware System Design

This section discusses the selection of hardware components that serve as the core elements of AI-driven IoT-based smart farming system. The hardware components are carefully selected as each component plays an important role in their respective distinct modules with specific functionality. Each component is carefully considered to ensure smooth integration, thus providing seamless operation flow of the whole system.

### 3.2.1 Raspberry Pi 4 Model B

In the context of designing an AI-driven IoT-based smart farming system, Raspberry Pi 4 Model B is implemented in the system to serve as a central processing unit (CPU). It is used to process and interpret data received from multiple sensors. It then further proceeds to execute instructions based on the data. The General-Purpose Input and Output (GPIO) pins offered by a Raspberry Pi 4 Model B allows it to interface with a wide range of peripherals. With this implementation, a surveillance system can be set up to monitor the presence of animals in the crop field. Besides, Raspberry Pi 4 Model B also plays its role in supporting IoT application in this system. This is because it provides ethernet port with various connectivity choices including Wi-Fi and Bluetooth, which makes it well-suited for communication and networking purpose. In simpler words, the ethernet port available in a Raspberry Pi 4 Model B is the key component for the connection of Raspberry Pi with other devices and routers (BasuMallick, 2022). Furthermore, Raspberry Pi 4 Model B offers high processor speed which boosts its performance. For these reasons, Raspberry Pi 4 Model B as shown in Figure 3.2 is chosen as part of the hardware system design. Table 3.1 shows the specifications of a Raspberry Pi 4 Model B (Raspberry Pi, 2019).

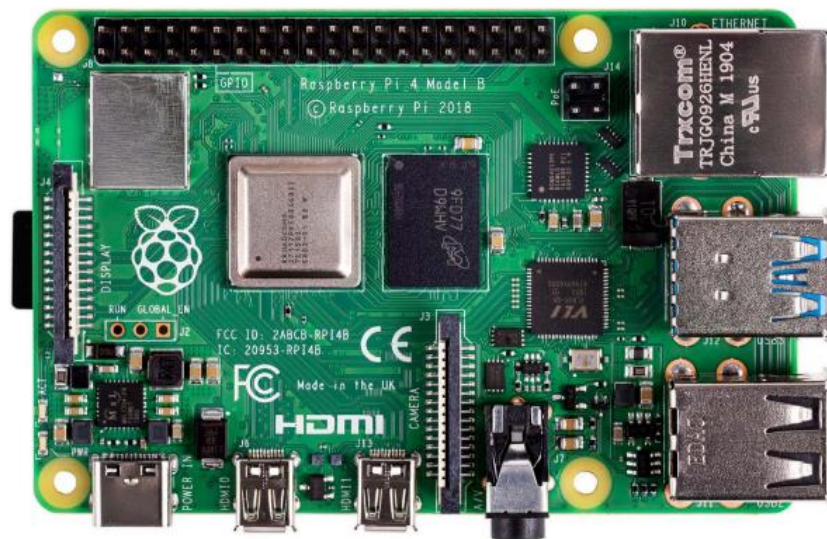


Figure 3.2: Raspberry Pi 4 Model B (Raspberry Pi, 2019).



Table 3.1: The Specifications of Raspberry Pi 4 Model B (Raspberry Pi, 2019).

Specification	Description
Processor	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5 GHZ
Memory	1 GB, 2 GB, 4 GB LPDDR4 (depending on model)
Connectivity	2.4GHz and 5.0GHz IEEE 802.11b/g/n/ac wireless LAN, Bluetooth 5.0, BLE Gigabit Ethernet 2 × USB 3.0 ports 2 × USB 3.0 ports
GPIO	Standard 40-pin GPIO header
SD card support	Micro SD card slot for loading operating system and data storage
Input power	5V DC

### 3.2.2 Raspberry Pi Camera Module v2

The camera module that is attached to the Raspberry Pi 4 Model B is known as Raspberry Pi Camera Module v2 as shown in Figure 3.3. As its name suggests, it is used to capture images and videos of the monitored area of the crop field. For this reason, it is acting as the “eyes” of the AI-driven IoT-based Smart Farming System to detect animal’s presence. Raspberry Pi Camera Module v2 consists of a Sony 1MX219 8MP sensor which allows it to capture high resolution images and videos (Raspberry Pi, n.d.). Besides, it also assures the quality of image captured, low-light performance and colour fidelity. It has a weight around 3 g which contributes to its compact and lightweight features. The specifications of Raspberry Pi Camera Module v2 is recorded in Table 3.2.

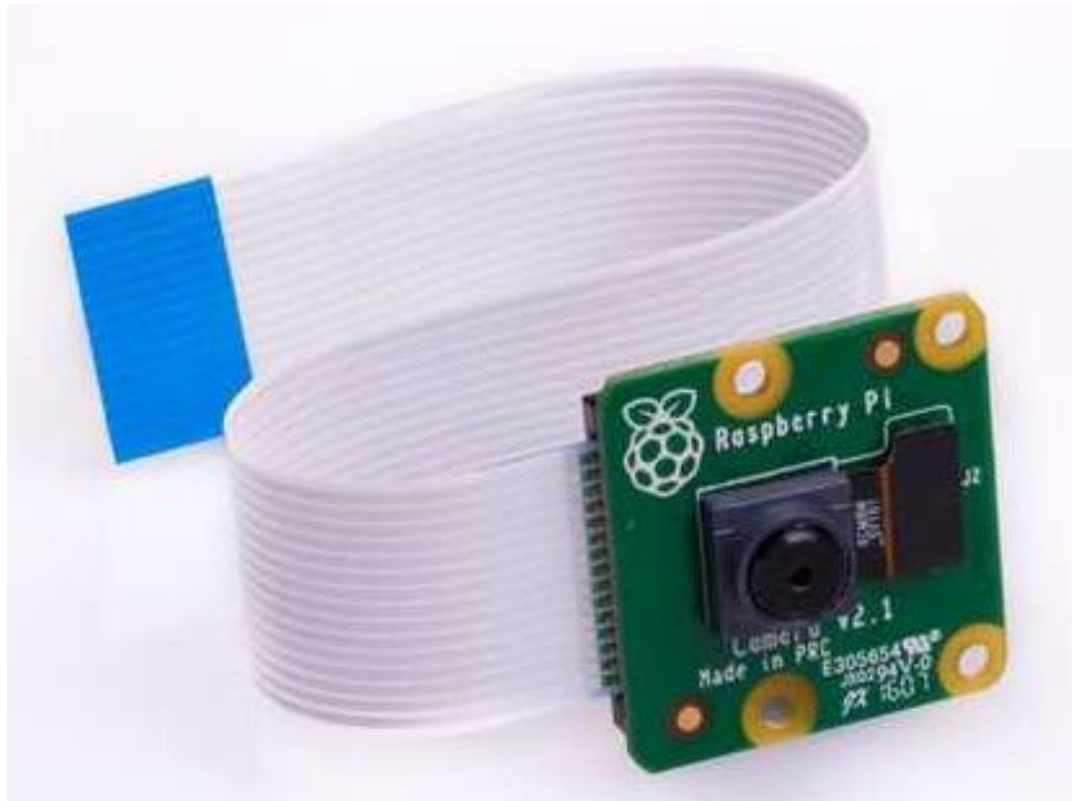


Figure 3.3: Raspberry Pi Camera Module v2 (Raspberry Pi, n.d.).

Table 3.2: The Specifications of Raspberry Pi Camera Module v2 (Raspberry Pi, n.d.).

Specification	Description
Dimension	25 mm × 23 mm × 9 mm
Weight	3 g
Resolution	8-megapixel
Lens	Fixed focus
Video modes	1080p30, 720p60, 640 × 480p90
Sensor	Sony IMX219

### 3.2.3 Passive Infrared Sensor

Figure 3.4 shows the Passive Infrared (PIR) Sensor that is being utilized and integrated in the system. PIR sensor is a motion sensor that aims to detect motion within the sensor range. It detects motion passively in such a way that it perceives the infrared radiation from objects that emit heat instead of emitting infrared signals by the sensor itself (Global Sensor Technology, 2021). This indicates that PIR sensor detects object motion via the movement of the object's infrared wavelength. In this proposed system, PIR sensor is integrated with Raspberry Pi Camera Module v2 to work together and produce an intelligent object detector. This is because PIR sensor acts as a master while Raspberry Pi Camera Module v2 acts as a slave. PIR sensor will trigger its slave which is the camera to activate and capture image only when it manages to detect cats. As a result, power consumption of the camera can be greatly minimized. Table 3.3 records the specifications of a PIR sensor (Ada and Dicola, n.d.).

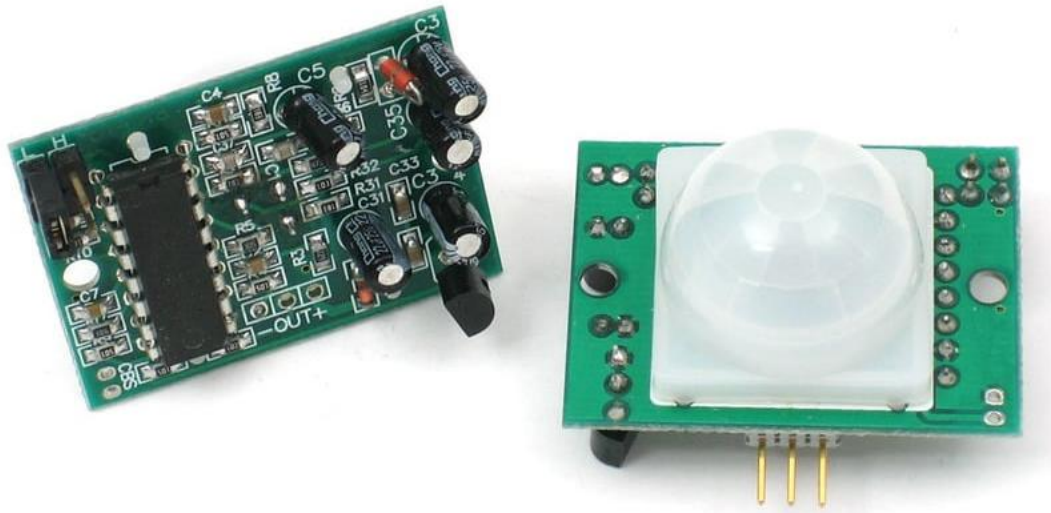


Figure 3.4: Passive Infrared Sensor (Global Sensor Technology, 2021).

Table 3.3: The Specifications of Passive Infrared (PIR) Sensor (Ada and Dicola, n.d.).

Specification	Description
Output	Digital pulse high when triggered (motion detected) Digital pulse low when idle (no motion detected)
Sensitivity range	Up to 7 meters, $110^{\circ} \times 70^{\circ}$ detection range
Power supply	5V – 12V input voltage

### 3.2.4 Buzzer

A buzzer as shown in Figure 3.5 is included in the proposed system to repel the intruding animals away from the monitored area by emitting ultrasonic sound. The model chosen is piezo buzzer as it can emit sound using piezoelectric effect that are unpleasant to some animals such as cats. In general, the emitted sound waves have frequency up to 5 kHz, which is considered sufficient enough to acts as a useful tool for causing a disruption to the auditory and nervous system of the targeted animals. Apart from this, this buzzer introduces a broad coverage where it can cover up to 309 square feet with a radius of around 9.84 feet.



Figure 3.5: Buzzer.

## 3.3 Software System Design

This section highlights the software system design that serves a primary purpose to orchestrate the functional logic of the smart farming system. The chosen programming language for the development of software programs, necessary software applications for system setup and IoT platforms are discussed in this section.

### 3.3.1 Python Programming

When it comes to the software design of the AI-driven IoT-based Smart Farming System, programming serves its purpose to develop a communication gateway between the hardware design and software design. Besides, programming is also needed for data processing, machine learning algorithms, decision making algorithms, database system development, and web server development. In this context, Python is the main programming language chosen for this proposed system. The logo of Python is shown in Figure 3.6. The main reason for choosing Python as the programming language is because it provides a strong and reliable platform for machine learning and artificial intelligence (AI). This is because Python comes with several machine learning libraries such as PyTorch, TensorFlow, Scikit-learn, and others that provides easy-to-use machine learning algorithms (Corporate Finance Institute Team, 2023). Besides, Python is independent across different platforms including Raspberry Pi, Linux and Windows. Python is also the preferred programming language as it offers great readability and simplicity in terms of its programming code. In fact, Python codes are relatively shorter and simpler than other programming languages. Thus, the execution speed of the program code is also faster.



Figure 3.6: Logo of Python (Ra20Ga, 2021).

### 3.3.2 VNC Viewer

Virtual Network Computing (VNC) viewer is an essential software application that is used with Raspberry Pi 4 Model B to access the Raspberry Pi's graphical desktop interface. Its logo is shown in Figure 3.7. VNC viewer is needed in this system as it enables remote accessibility of Raspberry Pi's desktop from a user's laptop, by connecting the VNC server and the Raspberry Pi over the same network. In other words, VNC viewer eliminates the necessity of using display monitor and external mouse.



Figure 3.7: Logo of VNC Viewer (Uptodown, n.d).

### 3.3.3 Telegram

In the context of designing AI-driven IoT-based Smart Farming System, Telegram is used as the software platform to develop the IoT-based monitoring system. Figure 3.8 shows the logo of Telegram. It functions to create an intuitive mobile interface which allows the users to receive alert notifications upon the animal invasions. Telegram is considered as a popular platform for IoT application development due to its versatility. It is versatile on various platforms and operating systems including IOS and Android. Moreover, Telegram is widely used in real-time applications as it supports media such

as images and videos. Besides, Telegram assures security because it only allows authorized users to access the system with the private bots.



Figure 3.8: Logo of Telegram (Logos-world, 2024).

### 3.4 Hard-Level Proposed System Design

In summary, Cropguard: AI-driven IoT-based Smart Farming System should be able to perform four main functions which are motion detection, image acquisition, repellent sound emission and alert notification system as shown in Figure 3.9. From the diagram in Figure 3.9, the Raspberry Pi 4 Model B acts as the main controller for the whole system, it is responsible for executing the entire software program. From another perspective, Raspberry Pi also serves as a IoT device and gateway to collect data from the input components and sensors such as Raspberry Pi Camera Module v2 and PIR sensor. Then it proceeds to perform data processing and transmit the data to the IoT cloud. Raspberry Pi is the edge device in this system to bridge the gap between the physical world and the digital world. In this case, Wi-Fi serves as a communication backbone to provide internet connectivity to the Raspberry Pi, allowing real-time integration and data transmission to the IoT cloud. On the other hand, buzzer will be activated according to the instruction received from Raspberry Pi, therefore it is the

output of the system. Meanwhile, Telegram plays its role in cloud-based communication by providing a platform for real-time alert notification.

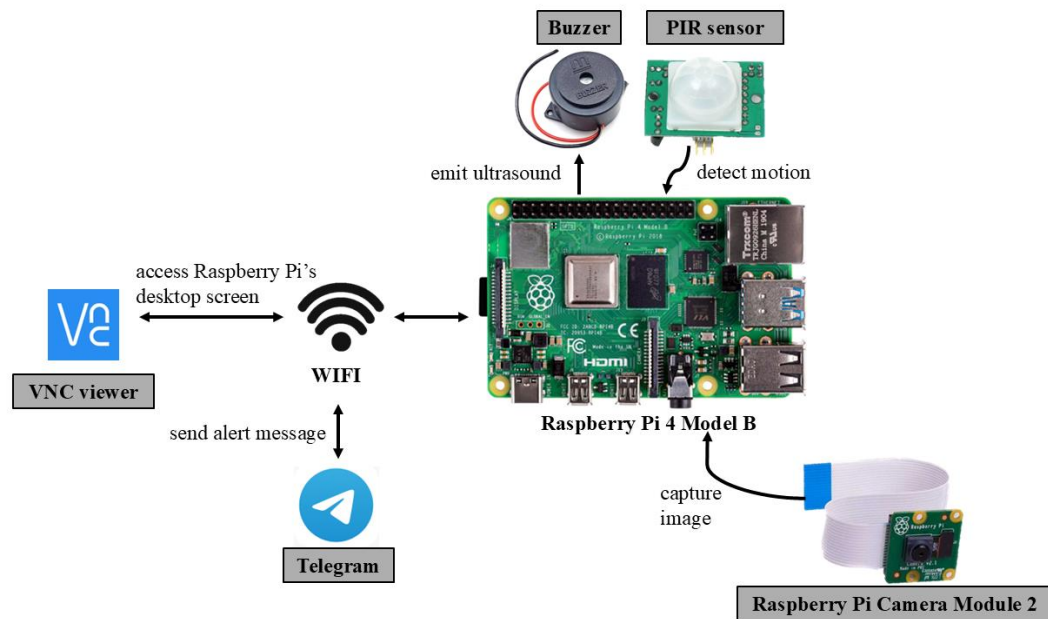


Figure 3.9: Hard-Level Design of the Proposed System.

### 3.5 Project Management

The project timeline spanning two trimesters are presented in two separate Gantt charts, breaking it down into several key phases and activities. The project for FYP 1 is progressed according to the timeline shown in Table 3.4. Generally, the project schedule for FYP 1 focuses on research and findings. Table 3.5 provides the project timeline for FYP 2, focusing on the development and implementation of the project. All in all, these Gantt charts are important for monitoring the progress of project, ensuring the project stays on schedule, and completing the project within project deadline.







## **CHAPTER 4**

### **RESULTS AND DISCUSSIONS**

#### **4.1 Hardware Development**

The hardware development of the smart farming system focuses on the integration of electronic components, the configuration and connectivity of all components, design and architecture of the whole prototype, including the design of a protective and adjustable-angle enclosure. The hardware development also emphasizes on the structural performance in an outdoor setting that have limited resources such as power supply, network connectivity, and so on.

##### **4.1.1 Electronic Components Integration**

In this section, all the electronic components are connected and integrated into a physical prototype which functions as an AI-driven IoT-based smart farming system. In simple words, the physical hardware setup realizes the hard-level system design as discussed in Section 3.4 by converting theoretical models into a real-time operating system. In the context of designing a smart farming system, the location of the hardware setup will be in an outdoor setting. For this reason, the hardware setup must be a standalone device without any physical connection to a monitor or mouse. In other words, it should be remote controlled using laptop or smart phone in an outdoor environment. On this basis, the Raspberry Pi is connected to a laptop instead of a

monitor by default. An ethernet cable is used to connect the Raspberry Pi directly with the laptop, allowing a direct communication gateway between these two devices. At the same time, Virtual Network Computing (VNC) server serves as a platform for the remote access of the Raspberry Pi's graphical desktop interface as shown in Figure 4.1. In this case, the VNC server which connects with the Raspberry Pi must have the same Wi-Fi network with the laptop itself to ensure that the Raspberry Pi's graphical desktop interface can be accessed using laptop.

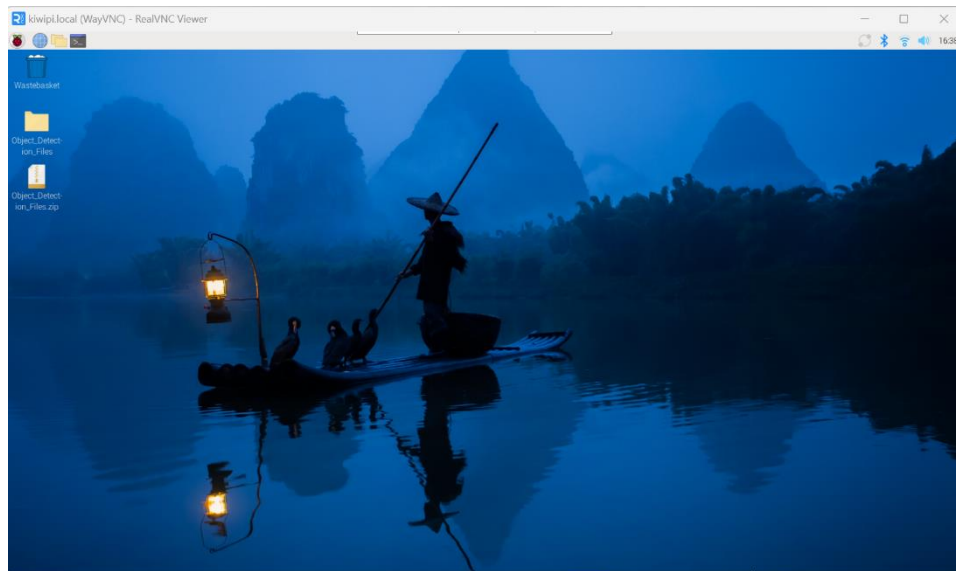


Figure 4.1: The Raspberry Pi's Desktop Interface through VNC Viewer.

#### 4.1.2 Enclosure Fabrication

To assembly and position all the electronic components and IoT device in an orderly manner, an enclosure for the whole prototype is customized and fabricated using 3-Dimensional (3D) printing technology. 3D printing technology contributes significantly to a customized-design enclosure that can fit all the electronic components precisely and house them in their respective positions and angles. From the perspective of exterior design, 3D printing enclosure improves the prototype's visual appearance and its overall presentation in a professional way. As its name suggests, 3D printing is a process of making 3D solid objects from file where the 3D product is printed out layer by layer. The design process starts with modelling software

which is responsible for the design of the shape and size of the enclosure. The 3D modelling software used for the enclosure are Thingiverse, SolidWorks and ideaMaker. In SolidWorks, tasks involving definition and adjustment of shape, size and dimension of the 3D printed enclosure are done. Besides, more particular tasks that delve into the finer details of the specifications, must be done precisely to ensure that the 3D printed enclosure can be fabricated well. For instance, infill, object density, temperature of the model, and others. The next step moves on to the hardware part of the 3D printing technology where the 3D enclosure is printed physically. 3D printer and filament are the primary elements of the fabrication process in terms of hardware. Before initiating the hardware printing process, the printer settings must be configured properly to ensure a smooth and successful printing. Printer bed must be adjusted to its optimal level according to the requirements of the 3D enclosure models. Besides, the functionality of different 3D printer parts such as nozzle, extruder, motherboard, and others must be checked properly. After all, the printing process is initiated, and the filament will be heated and melted to print the enclosure models layer by layer from the bottom to the top. Figure 4.2 shows the configuration setting of enclosure for Raspberry Pi Camera Module v2 in raise 3D printer. Figure 4.3 presents the current progress of 3D printing for the support structure for the prototype. There are a total of four 3D enclosure models that are needed to be fabricated to protect and secure all the electronic components. For instance, Raspberry Pi 4 Model B, Raspberry Pi Camera Module v2, Passive Infrared (PIR) sensor and buzzer. After the fabrication of these enclosures have completed separately, an assembly of all these enclosures is required in the next step. This is essential to ensure that each functional components are positioned in their respective optimal angles and the whole prototype is presented professionally. Since Raspberry Pi Camera Module v2 work with the assistance of PIR sensor, therefore these two components will be placed next to each other. Therefore, a customized design that is tailored to the specifications of the placement are required. Figure 4.4 provides the design preview of the assembly enclosure for both components in ideaMaker software. Besides, Figure 4.5 illustrates the design preview of Raspberry Pi 4 Model B's enclosure in SolidWorks.



Figure 4.2: 3D Printer Home Setting for Raspberry Pi Camera Enclosure.

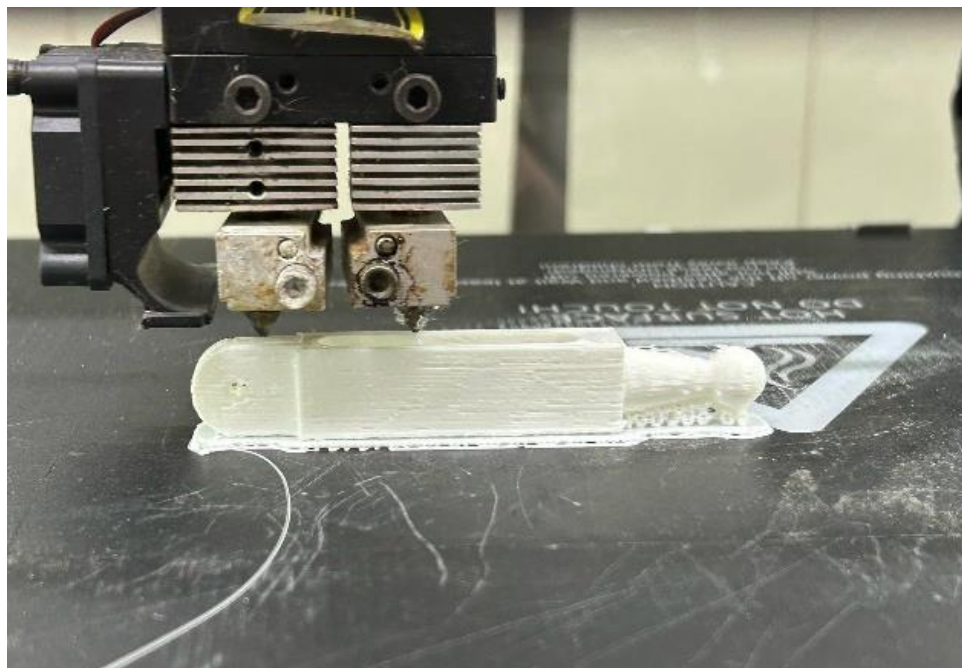


Figure 4.3: 3D Printing for the Support Structure of the Prototype is in Progress.

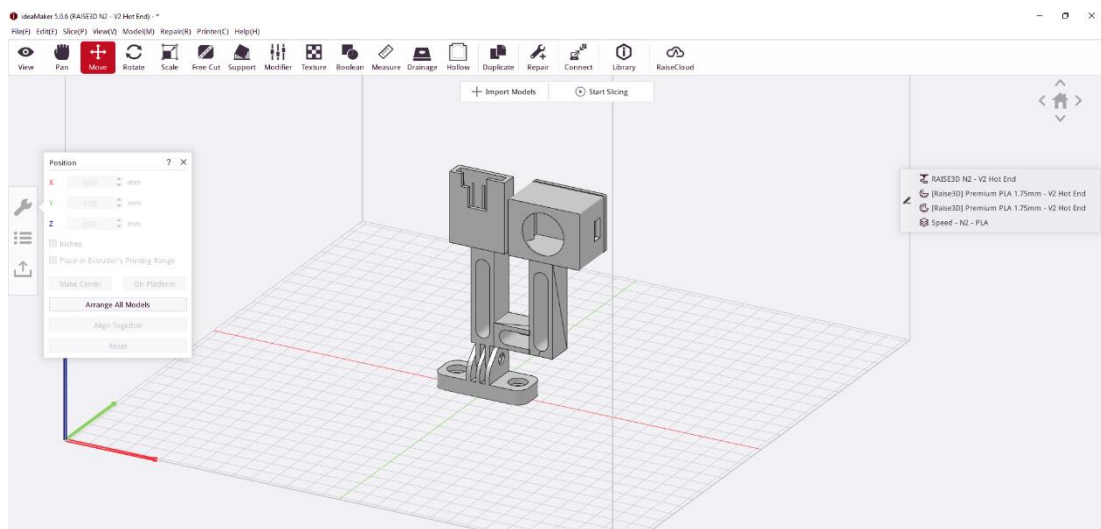


Figure 4.4: Design Preview of Assembly Enclosure for Raspberry Pi Camera Module V2 and PIR Sensor in IdeaMaker Software.

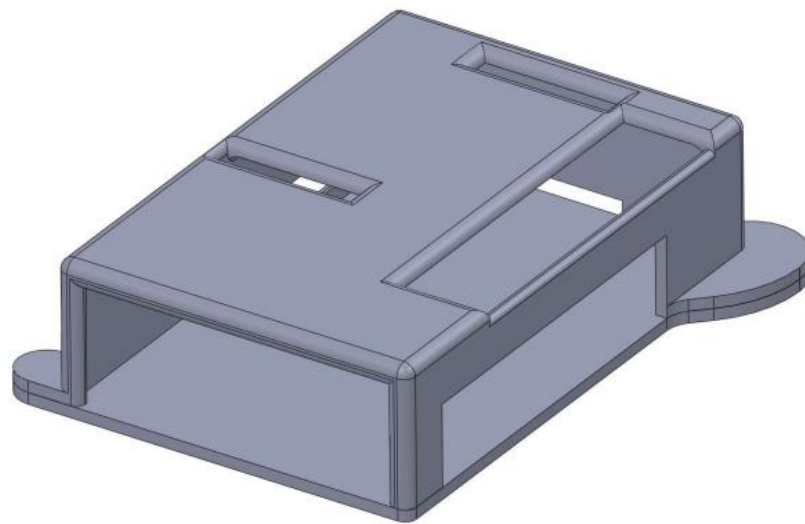


Figure 4.5: Design Preview of Enclosure for Raspberry Pi 4 Model B in SolidWorks.

### 4.1.3 Prototype

Figure 4.6 shows the complete prototype of this project with all functional components securely mounted and held in place. By referring to Figure 4.6, the Raspberry Pi Camera Module v2 is set up side by side with the Passive Infrared (PIR) sensor since

both components operate collaboratively. Pi camera works with the assistance of PIR sensor, Pi camera will only be activated when PIR sensors is triggered by motion. If PIR sensor is triggered, this indicates that motion is detected in front of the Pi Camera, thus it is necessary to place the Pi camera beside the PIR sensor so that the Pi camera can capture the exact view of the detected motion. The sensitivity range of PIR sensor is 3 to 7 meters which means that it can detect motion up to 7 meters. If there is any motion takes place beyond 7 meters away from the PIR sensor, the motion will not be detected by PIR sensor. In addition, the detection angle of a PIR sensor spans  $120^\circ$ , this indicates that its effective coverage is  $60^\circ$  to the left and  $60^\circ$  to the right from its central axis. The horizontal field of view of Raspberry Pi Camera Module v2 is approximately  $62.2^\circ$ , which implies that Pi camera has a total horizontal width of capture angle from  $31.1^\circ$  to the left and  $31.1^\circ$  to the right of the central axis. On the other hand, its vertical field of view is approximately  $48.8^\circ$ , indicating that the vertical capture angle spans  $48.8^\circ$ , ranging from  $24.4^\circ$  upward and  $24.4^\circ$  downwards from the central axis of Pi Camera. Besides, the buzzer is placed on top of the PIR sensor, facing the same direction with PIR sensor. This is to ensure that the sound produced by the buzzer can effectively repel the cat from its exact opposite direction. In other words, the repelling action is executed directly in front of the cat detected. Furthermore, a 20,000 mAH power bank serves its purpose to acts as an independent power supply for the system. The power bank is connected to the Raspberry Pi to provide sufficient power to the system so that the whole system can acts as a standalone device. In terms of duration, the power bank is capable to supply power to the Raspberry Pi for about 12 to 13 hours.



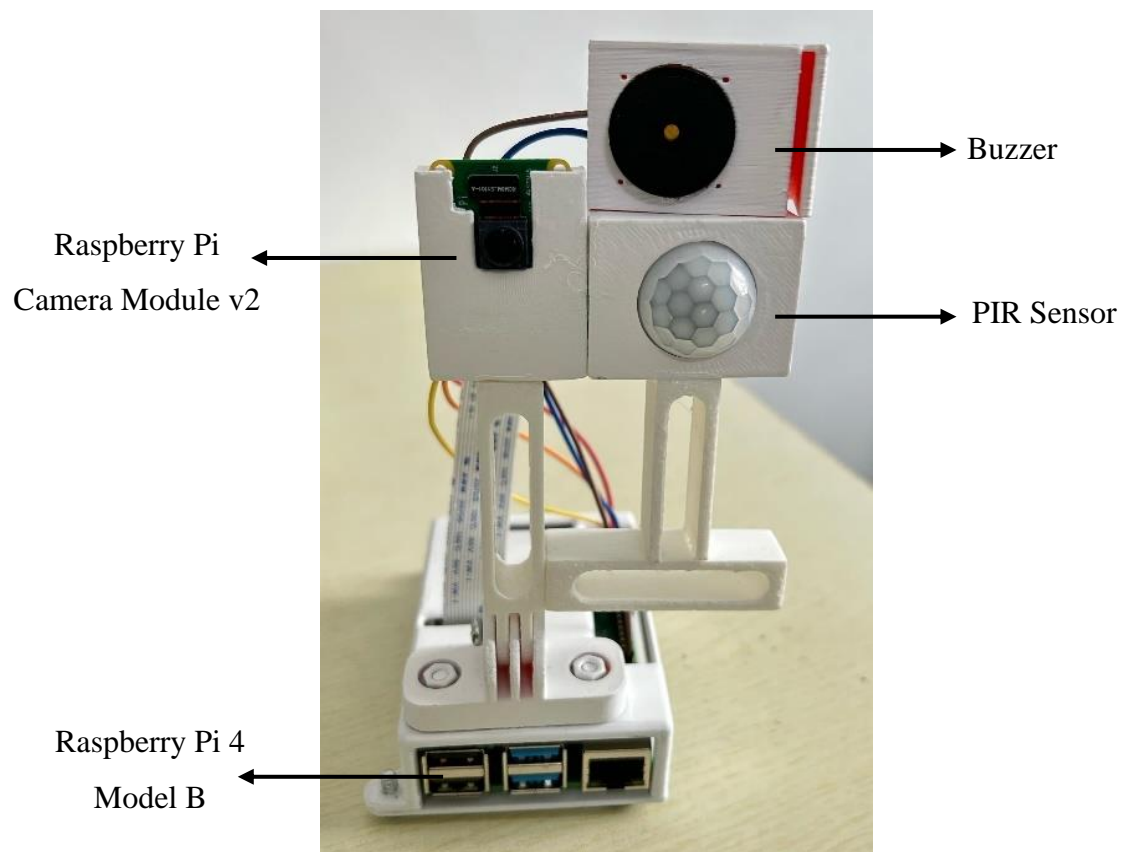


Figure 4.6: Prototype of the Smart Farming System.

## 4.2 Software Development

The software development of the smart farming system focuses on the implementation of functional logic for different modules with their respective specific functionalities. Several key aspects including energy efficiency, lightweight computation, compactness, accuracy, consistency and reliability are the highlights in this section.

### 4.2.1 Software Setup

When it comes to IoT projects, operating system is the fundamental element to manage hardware development and support software development, allowing applications to

communicate with the hardware and other system-level functions. In this project that involved Raspberry Pi, the operating system used is called Raspberry Pi OS. Besides, VNC viewer is another software application used in this project to allow direct access between laptop and the Raspberry Pi desktop without any external support devices such as monitor and mouse. In the initial stage of software development, all necessary configuration tasks for each component are implemented. For instance, functionality testing, setup and initialization, software and packages installation needed to be done before developing the software part of the smart farming system. Figure 4.7 shows that the installation for OpenCV is successfully performed in the terminal window of Raspberry Pi OS.

```
(tflite-env) pi@kiwipi:~/tensorflow-yolov4-tflite $ pip install opencv-python-headless==4.7.0.72
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting opencv-python-headless==4.7.0.72
  Using cached opencv-python-headless-4.7.0.72.tar.gz (91.1 MB)
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: numpy>=1.21.2 in /home/pi/tflite-env/lib/python3.11/site-packages (from opencv-python-headless==4.7.0.72) (1.24.0)
Building wheels for collected packages: opencv-python-headless
  Building wheel for opencv-python-headless (pyproject.toml) ... done
  Created wheel for opencv-python-headless: filename=opencv_python_headless-4.7.0.72-cp311-cp311-linux_aarch64.whl size=11969997 sha256=54b7c98128c0993531678c01bc5200db1f9401c7450ecfbf512af6a4af132527
  Stored in directory: /home/pi/.cache/pip/wheels/f4/a1/6b/8dcdf604e80f240c9385a271aae54d9d8670e83223fb46f62
Successfully built opencv-python-headless
Installing collected packages: opencv-python-headless
  Attempting uninstall: opencv-python-headless
    Found existing installation: opencv-python-headless 4.11.0.86
    Uninstalling opencv-python-headless-4.11.0.86:
      Successfully uninstalled opencv-python-headless-4.11.0.86
Successfully installed opencv-python-headless-4.7.0.72
```

Figure 4.7: Successful Installation of OpenCV library.

#### 4.2.2 Classification Model

The core aspect of this project revolves around the AI techniques employed for object detection, specifically focusing on cats. On this basis, a machine learning model is used to classify animals whether they are cats or not. In this case, a machine learning model is pre-trained using OpenCV in such a way that a large dataset of cat images is being fed into the model for it to learn about their common pattern from the dataset. With this, the model will analyze the pattern and is capable of implementing predictions on new data based on the pattern learned. Concerning the practical

limitations of Raspberry Pi imposed by various factors such as limited random-access memory (RAM), limited memory capacity and power intensive, a machine learning algorithm which is known as Haar Cascade classifier is employed in smart farming system for the purpose of object detection. As its name suggests, Haar Cascade classifier is a machine learning algorithm that uses a cascade of classifiers to detect object based on the Haar-like features. In this context, OpenCV provides a pathway for Haar Cascade classifier with the objective of implementing real-time object detection. The first step in training Haar Cascade classifier for cats is collecting a large dataset of positive images of targeted subjects and negative images of non-targeted subjects. In simpler words, those positive images will be different kinds of cats' images while negative images can be images other than cats. In this context, there are thousands of positive images and negative images that have been collected for the purpose of training. Next, it is essential to indicate the location of the targeted subjects inside every positive image, thus the coordinates of the bounding boxes that precisely locate the targeted subject should be prepared and recorded in a list file. Meanwhile, since the negative images do not have the targeted subjects, hence there is no need to record the precise location of the non-targeted subjects. After all the dataset of positive images and negative images have been collected, the dataset is ready for training. The dataset is being fed into a training tool that comes with the OpenCV library. During this stage, the training tool will train the classifier model to apply and calculate Haar-like features based on the collected dataset. In general, it is trained to identify the common patterns that are frequently present in the positive images of targeted subjects like cats and extract the common features from the images. For instance, the eyes of a cat will be recognized as a darker region while the nose of a cat will be recognized as a brighter region. Therefore, the Haar-like features are collected and will be proceeded for calculation. During this stage, the addition of pixel intensities in each region will be calculated. At the same time, the subtraction of the sum of pixels is also calculated. However, Haar-like features can be difficult for big image, therefore integral image is being introduced to reduce the complexity of calculations. In other words, integral images help to compute the Haar-like features by generating sub-rectangles and array references for each of the sub-rectangles (Mittal, 2020). Figure 4.8 clearly illustrates the concept of integral images for Haar-like features. The next stage involves AdaBoost training where it is a boosting technique used to select the key features that

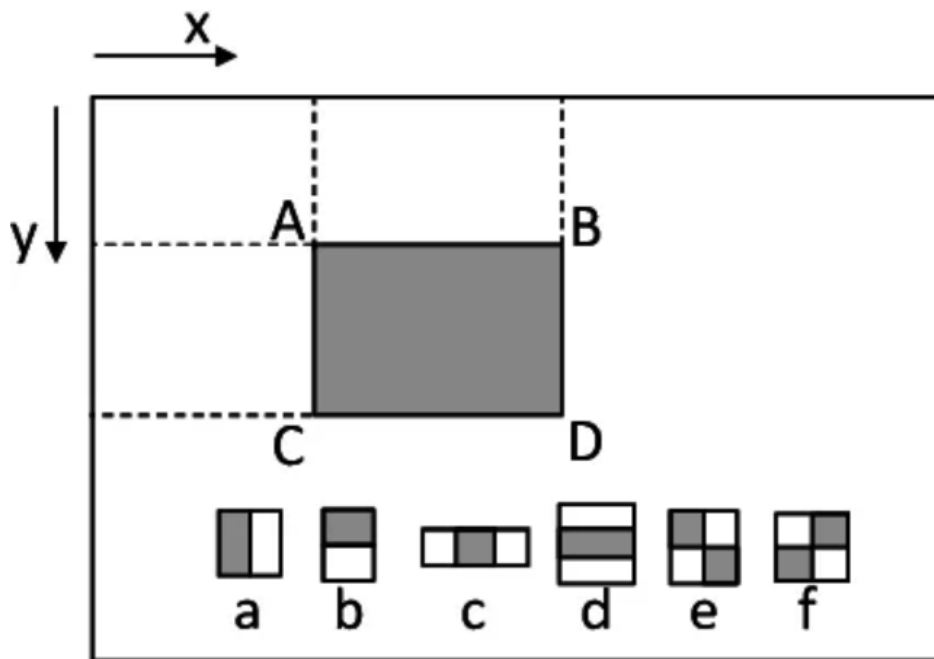


Figure 4.8: Working Concept of Integral Images for Object Detection (Mittal, 2020).

```

<!-- stage 1 -->
<_>
  <maxWeakCount>26</maxWeakCount>
  <stageThreshold>-1.4618960618972778e+00</stageThreshold>
  <weakClassifiers>
    <_>
      <internalNodes>
        0 -1 725 1.0133055038750172e-02</internalNodes>
      <leafValues>
        -2.8207325935363770e-01 6.2703561782836914e-01</leafValues></_>
    <_>
      <internalNodes>
        0 -1 356 3.8468956947326660e-02</internalNodes>
      <leafValues>
        -1.4483113586902618e-01 7.4971008300781250e-01</leafValues></_>
    <_>
      <internalNodes>
        0 -1 2 -3.7523733917623758e-03</internalNodes>

```

Figure 4.9: Part of the Code to Classify Cats based on Different Sets of Parameters.

best distinguish a cat from a huge dataset of Haar-like features. AdaBoost plays an important role in enhancing the accuracy of a classifier model because it forms a strong classifier by combining several weak classifiers. In other words, a strong classifier can be considered as a weighted sum of the weak classifiers. The final stage in training this machine learning model is to proceed with the cascading classifiers. This stage involves several stages of classifiers where all the features of a cat are split into various classifiers separately. The classifiers stage is then applied individually, step by step. By way of example, the first classifier stage is a checkpoint for feature like cat's eyes. If the image window is positive and matches with the first feature, then it will proceed to the next classifier stage with different features. On the contrary, the image window will fail at the first classifier stage if the image is negative. Hence, this window will be rejected and will not be considered for the next cascade classifier stage. After all, if a window manages to pass all the cascade classifier stages, it can be said with certainty that this window presents a cat. Figure 4.9 shows extracted part of the data for Stage 1 Cascade Classifier stage in .xml file. Based on the extracted data, stage 1 is composed of 26 weak classifiers and the threshold value in stage 1 is  $-1.4618960618972778e+00$ . If the result in stage 1 is greater than  $-1.4618960618972778e+00$ , then it can proceed to the next stage and vice versa. Each group represents one weak classifier with different sets of parameters. For the first weak classifier, 0 -1 725 1.0133055038750172e-02 at internal nodes represents the parameters for features in the first weak classifier whereas the leafValues which are  $-2.8207325935363770e-01$  and  $6.2703561782836914e-01$  represent the output results. After all the training has been completed, the machine learning model is well-trained as a Haar Cascade classifier, and it is saved as 'haarcascade\_frontalcatface.xml' file.

### 4.3 Unified System Workflow

This section highlights the operational cycle of the complete system by breaking it down into two subsections. The first subsection, which is system-level operation workflow, emphasizes the integration between hardware and software. Integration is significant to enable interaction between the hardware components. This subsection

discusses the system's physical operation. On the other hand, the second subsection, which is software logic and execution flow narrow down to discuss the structure of the software program logic. It explains the digital brain behind the system, focusing on the internal structure of the system.

### **4.3.1 System-level Operation Workflow**

In the context of designing an AI-driven IoT-based smart farming system for agricultural farming, electricity supply is one of the factors that should be taken into careful consideration. This is because the agricultural sector often faces challenges in obtaining sustainable electricity supply. Therefore, this underscores the importance of developing a smart farming system that is energy efficient to tackle the common challenges faced in the agricultural sector. This project stands out for its remarkable performance in achieving a sustainable electricity supply by adopting several measures to save energy. Figure 4.10 gives an illustration of the flowchart of the smart farming system. This project develops a cutting-edge approach by introducing an intelligent system that has bistable mode control to reduce unnecessary energy consumption. This system is designed in such a way that the Pi camera will only be switched on when motion is detected by the PIR sensor or else the Pi camera will be in idle mode. If motion is detected, the PIR sensor will trigger the camera module to switch on but there is no image acquisition until a cat is detected by Haar Cascade classifier. In other words, the camera will be activated automatically to act as an eye to observe the surrounding environment, it will capture image only when the system detects there are cats present in the surrounding environment. This signifies a holistic approach towards an intelligent and energy-efficient system. This is due to the fact that this system does not simply keep the camera module turned on constantly, but it is only powered on under specific conditions. With this approach, the camera is used only, when necessary, thus the camera energy can be conserved. Besides, image acquisition will only be performed for cats. When the Pi camera captures images, the images will be renamed according to the present time when it has been captured and saved in a folder named as 'detected\_cats' in Raspberry Pi as shown in Figure 4.11. Several alert actions

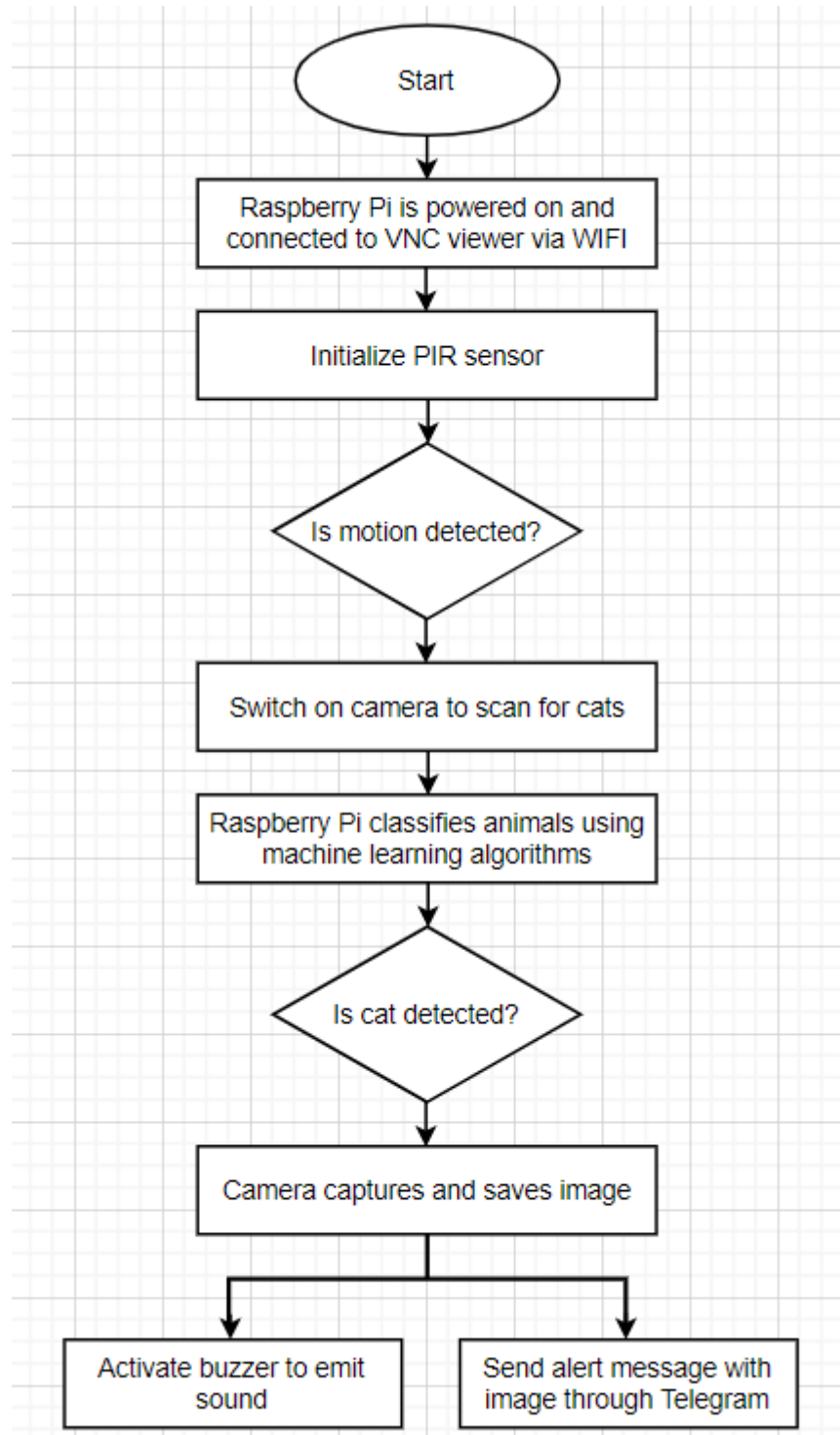


Figure 4.10: Flowchart of Smart Farming System.

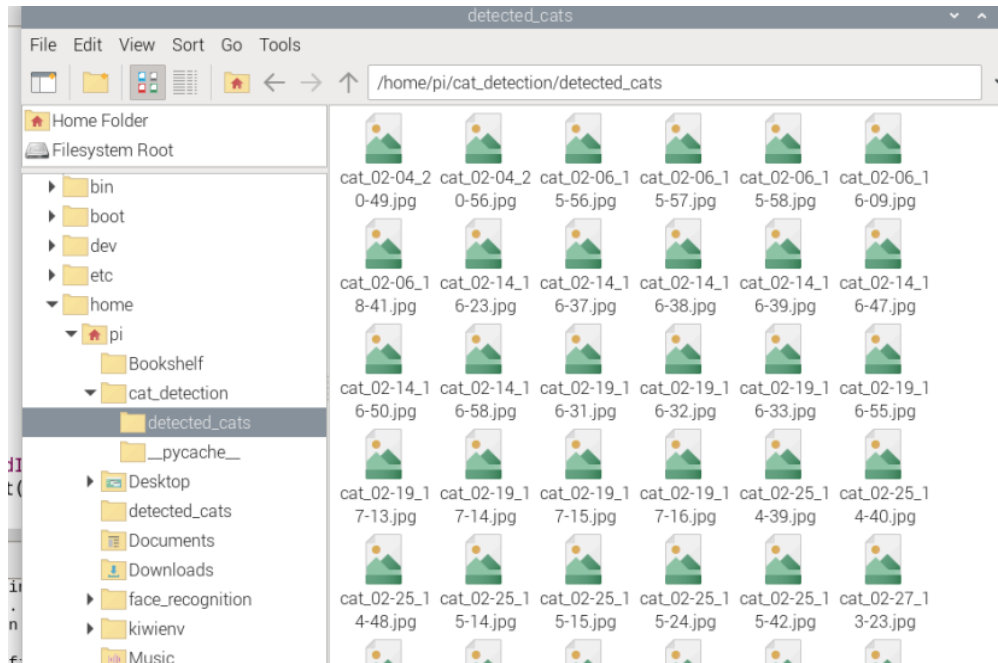


Figure 4.11: Images Saved in Folder.

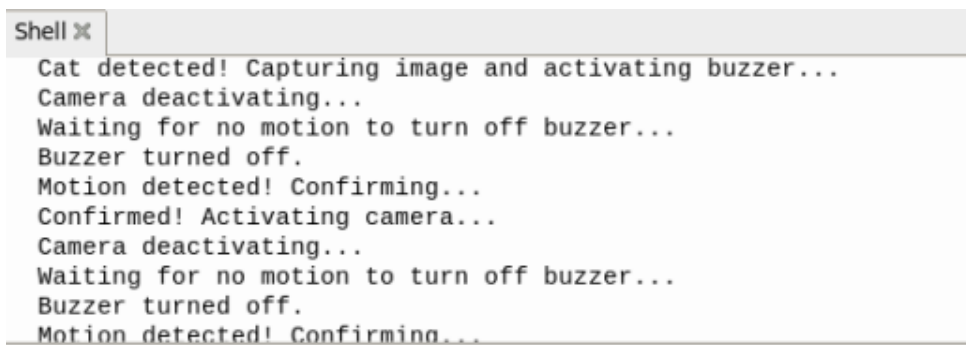


Figure 4.12: Message in the Shell of System Program.



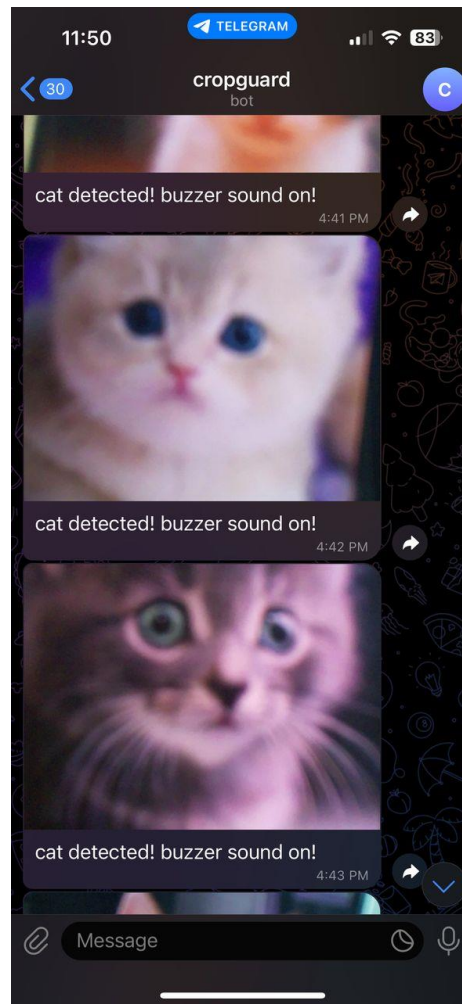


Figure 4.13: Alert Message in Telegram.

including activation of buzzer alarm and alert message sending will be triggered simultaneously and immediately when cat is detected and captured by the system. The buzzer will be activated to produce sound for a minimum duration of 5 s and stop producing sound only when the cat moves away from the detection range of the system. The buzzer will produce continuous alarm output as long as the detected cat is still present within the detection range of the system. In other words, the duration of the buzzer's sound emission depends on the state of the PIR sensor. Figure 4.12 shows the status of the smart farming system which clearly indicates every activation status of the functional devices. At the same time, alert messages which include the image of cat detected will be sent immediately to the user through Telegram as shown in Figure 4.13.

### 4.3.2 Software Logic and Execution Flow

In the beginning, it is necessary to open VNC viewer so that users can access the Raspberry Pi's desktop screen and proceed to develop the software program in the Raspberry Pi. PuTTY is an application that needs to be used to open and set up the VNC viewer in the condition when no monitor is connected to the Raspberry Pi. PuTTY establishes Secure Shell (SSH) connections between a laptop window and external devices like Raspberry Pi. Figure 4.14 shows the PuTTY configuration before running VNC viewer. Figure 4.15 shows the authentication of VNC server before getting into VNC viewer. Appendix A presents the full program script for smart farming system which is written in Python language. The python script starts with the modules imports to perform initial setup of the program environment. It is important to import the OpenCV library for computer vision and image processing that are required in this system. Apart from that, the requests module is imported to allow communication between the software program and the telegram application programming interface (APIs). This enables the message and data to be sent to Telegram over the Internet. Other modules including file handling modules, RPI.GPIO modules, Picamera2 modules are required to develop the integrated unified system. The next phase of software logic and execution flow is to configure the setup of Telegram bot. At first, a telegram bot with a specified name 'cropguard' is created through BotFather in Telegram. Then, a Bot Token will be sent in the chat to access the HyperText Transfer Protocol (HTTP) API. The Bot Token received is 8115057421:AAHYv-nEQ7nnLSFs-iWxG-uUChD3DgbtGwg. The subsequent step is to obtain the chat id for 'cropguard' by initiating a new conversation with the new bot 'cropguard' and extracting the chat id provided in the Telegram API browser. The obtained values of both bot token and chat id are specified in the script to configure the Telegram, allowing a secure alert message notification. The python script continues with the initialization and setup of the GPIO pins for Raspberry Pi 4 Model B. The GPIO pin 4 of Raspberry Pi is configured as the input for PIR sensor while the GPIO pin 17 of Raspberry Pi is configured as the output for buzzer. Next, the python script also introduces logging event functions to record and track every event of the system with detailed information such as the live status of every functional component that comes along with the current date and time as shown in Figure 4.16. The detection

log file is used for performance tracking. Next, the pre-trained machine learning model, Haar Cascade classifier that was generated beforehand is imported within the script. In this context, OpenCV module serves a purpose to load the Haar Cascade classifier in eXtensible Markup Language (XML) format for object detection and classification. When Raspberry Pi executes the 'haarcascade\_frontalcatface.xml' during Pi camera activation, Haar Cascade classifier will perform continuous scanning of every live frame within the camera. Figure 4.17 and Figure 4.18 show the process of converting the input frame to grayscale, respectively. Once the input frame is converted to grayscale, a window with a size of  $40 \times 40$  will slide across all positions of the grayscale frame to check every aspect of cat's features. The diagrams from Figure 4.19 to Figure 4.24 shows the  $40 \times 40$  window at different positions of the grayscale frame. After obtaining windows at different positions, this is when the cascade classifier stages come into play to detect whether the frame is a cat. The classifiers stages are applied individually, step by step to every window. Each classifier stage is responsible for checking different aspects of features related to cats. If a window succeeds in passing all the cascade classifier stages, then this input frame is considered as a cat. Figure 4.25 shows that a cat is successfully detected by the classification model. There is a bounding box that encompasses the area of cat detected. In addition, the Pi camera is initialized to have a minimum active time of 10 s once PIR sensor detects motion. This is to ensure that the motion-triggered camera has enough time to monitor and scan the live frame of the motion detected. With this, the accuracy of object detection can be further enhanced. Apart from that, stabilization of the PIR sensor plays a significant part in ensuring accurate motion detection to minimize false positives. In this case, the PIR sensor is stabilized for 30 s after the system is powered on and ready for operation. In the first 30 s, the PIR sensor is allowed to make adjustment to the ambient temperature as it detects motion by perceiving infrared radiation from objects that emit heat. If the PIR sensor is not stabilized, it might detect false motion due to several possible reasons such as the heat caused by the setup and initiation of component power-up. Figure 4.26 illustrates the confirmation status of stabilization of the PIR sensor in the beginning of system program start-up. In the main program loop, PIR sensor acts as the input of the whole program since the activation of the rest of the functional components depends on PIR sensor. In simple words, PIR sensor is the master of the program while Pi camera, buzzer and Telegram

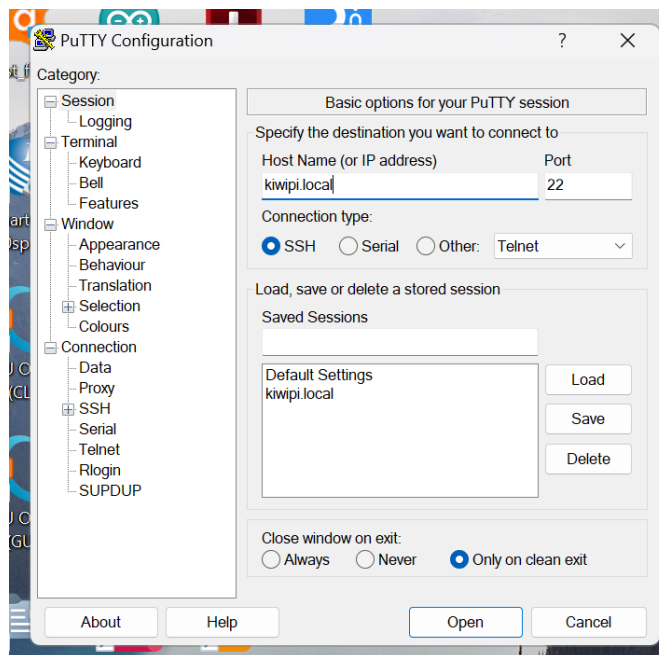


Figure 4.14: PuTTY Configuration.

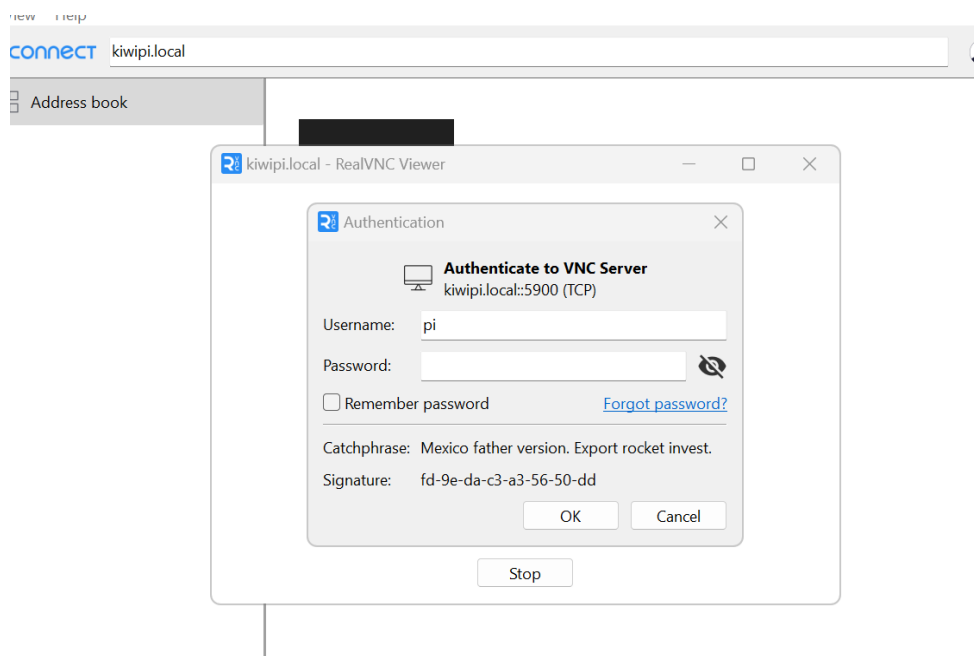


Figure 4.15: Authentication of VNC Server.

```

[2025-04-22 16:44:45] Motion detected
[2025-04-22 16:44:45] Camera deactivated
[2025-04-22 16:44:45] Buzzer deactivated No motion detected
[2025-04-22 16:44:48] Motion detected
[2025-04-22 16:44:52] Cat detected - Image saved as detected_cats/cat_1
[2025-04-22 16:44:52] Buzzer activated
[2025-04-22 16:45:01] Camera deactivated
[2025-04-22 16:45:01] Buzzer deactivated No motion detected
[2025-04-22 16:45:04] Motion detected
[2025-04-22 16:45:17] Camera deactivated
[2025-04-22 16:45:17] Buzzer deactivated No motion detected
[2025-04-22 16:45:25] Motion detected
[2025-04-22 16:45:32] Cat detected - Image saved as detected_cats/cat_1
[2025-04-22 16:45:32] Buzzer activated
[2025-04-22 16:45:37] Camera deactivated
[2025-04-22 16:45:37] Buzzer deactivated No motion detected
[2025-04-22 16:45:42] Motion detected
[2025-04-22 16:45:54] Camera deactivated
[2025-04-22 16:45:54] Buzzer deactivated No motion detected
[2025-04-22 16:45:57] Motion detected
[2025-04-22 16:46:10] Camera deactivated
[2025-04-22 16:46:10] Buzzer deactivated No motion detected
[2025-04-22 16:46:14] Motion detected
[2025-04-22 16:46:27] Camera deactivated
[2025-04-22 16:46:27] Buzzer deactivated No motion detected

```

Figure 4.16: Detection Log File for Event Tracking.



Figure 4.17: Input Frame.



Figure 4.18: Converted Grayscale Frame.

are the slaves that receive instructions from PIR sensor to operate. Upon motion detection by PIR sensor, the camera will be activated for at least 10 s, then Haar Cascade classifier comes into play to scan the live frame within the camera and detect if it is a cat. Once a cat is detected, there are several actions that will be initiated. For instance, the Pi camera will capture and save the image in 'detected\_cats' folder, the buzzer will produce sound to repel the cat, a message with an image of the cat will be sent to Telegram, and the latest detection time of cat will be updated in the detection log file. Once the motion is no longer present within the detection range of PIR sensor, the buzzer will turn off and stop producing sound. The Pi camera and buzzer will reset and wait for instruction from the PIR sensor until motion is detected again.

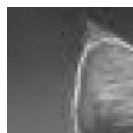


Figure 4.19: Window at Position (1,24).



Figure 4.20: Window at Position (60,60).



Figure 4.21: Window at Position (100,100).



Figure 4.22: Window at Position (130,50).



Figure 4.23: Window at Position (40,100).

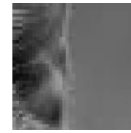


Figure 4.24: Window at Position (30,150).

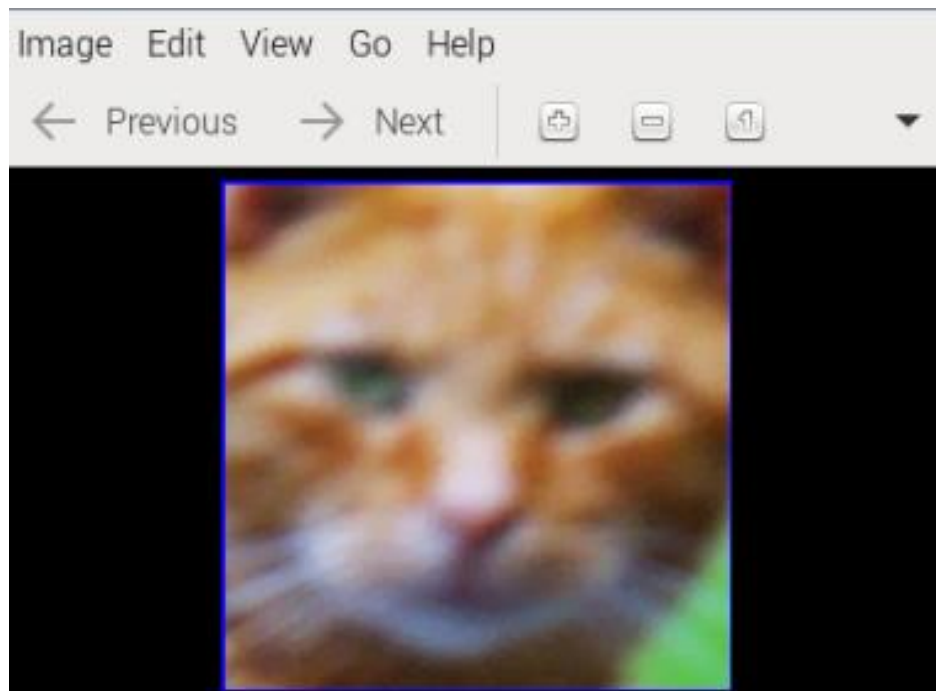


Figure 4.25: A cat is Successfully Detected.

```
Initializing PIR sensor... Waiting for stabilization (30s)  
PIR sensor ready.  
Waiting for motion... (Press Ctrl+C to exit)
```

Figure 4.26: Stabilization of PIR Sensor.

## **4.4 System Analysis and Evaluation**

In this section, the performance of the AI-driven IoT-based Smart Farming System is being analyzed and evaluated using several key measures which will be discussed in the following subsections. System analysis and evaluation are one of the important phases in the development process of the system to ensure that the system achieves performance milestones required by the system. It examines the effectiveness, accuracy, and reliability of the system under several conditions.

### **4.4.1 Detection Performance Analysis**

As its name implies, detection performance analysis focuses on the evaluation process to determine the accuracy and effectiveness of the smart farming system in detecting and classifying cats. This subsection highlights the performance of the detection part of the system. In this context, several performance metrics such as confusion matrix, recall, precision, F1-score and detection latency are conducted. Through detection performance analysis, the strength and weakness of the classification model in detecting cats can be identified and analyzed for future improvements.

#### **4.4.1.1 Confusion Matrix Analysis**

Confusion Matrix is a method to evaluate the performance of the machine learning classifier by identifying the classification accuracy, breaking it down into four different categories. By referring to Figure 4.27, the top-left cell of the confusion matrix refers to the True Positives (TP) cases, that is the case where positive detections are obtained. In simpler words, TP case refers to the case where the system correctly predicted the presence of a cat when a real cat was actually present. This indicates that a cat is successfully detected by the system. For this reason, the system is considered highly sensitive in detecting cat, since the system demonstrates an accurate detection



performance. The top-right cell of the confusion matrix refers to the False Negatives (FN) cases where the system fails to detect the presence of a real cat. The system could not detect the presence of cat therefore cause a missed detection. The FN rate obtained from the system is 4.55% which is considered relatively low. Apart from that, the bottom-left cell of the confusion matrix represents the False Positive (FP) rate of the system. False positive case refers to the case where the system performs false detection, it predicted the animal as a cat, but the animal is not a cat in actual case. Non-cat is detected by the system, leading to a false deterrence response to the animal that is not a cat. Figure 4.27 shows that the system has a FP rate of 9.09% which indicates that the 9.09% of all the cases were misclassified as cats. Lastly, the bottom-right cell of the confusion matrix represents the True Negative (TN) rate of the system. It refers to the correct detection performed by the system in negative cases where cat is not present but other animals are present. In other words, non-cats are not detected by the system. The system shows a high TN rate of 45.45% that reflects the remarkable performance of the system in eliminating irrelevant inputs fed to the system. When other animals such as dog, tiger, kangaroo, and so on are present, the system will not detect them as cats. The overall accuracy of the system's classification model is calculated using Equation 4.1.

$$Accuracy = \frac{TP + TN}{Total\ cases} \times 100\% \quad (4.1)$$

where

$TP$  = True Positive

$TN$  = True Negative

As a result, the overall accuracy obtained is 86.36%, which is considered relatively high. This accuracy result reflects the system's ability in detecting and classifying cats accurately and consistently. The system has shown great potential in minimizing crop damage due to its strong and stable performance.

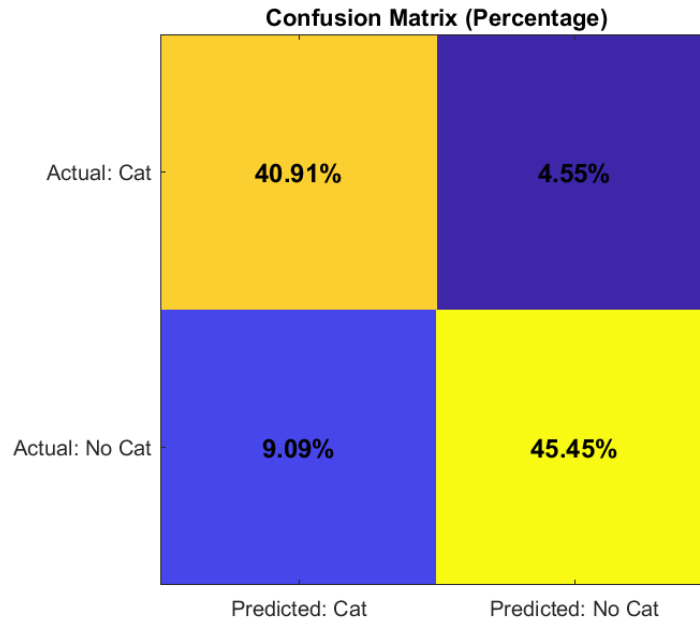


Figure 4.27: Confusion Matrix of the Classification Model.

#### 4.4.1.2 Performance Metrics Evaluation

Besides, the performance of the detection module is evaluated using another three key metrics which are recall, precision and F1-score. Figure 4.28 shows the value of recall, precision and F1-score which are 89.99%, 81.82% and 85.71% respectively. Recall is used as a metric to measure how many cats are actually detected. It can be calculated using the value of TP and FN as shown in Equation 4.2.

$$Recall = \frac{TP}{TP + FN} \quad (4.2)$$

where

$TP$  = True Positive

$FN$  = False Negative

Since the value of  $TP$  and  $FN$  obtained from the confusion matrix as shown in Figure 4.27 are 40.91% and 4.55%, respectively, therefore the result of recall is 89.99%. This result implies that out of all the actual cats that are present, 89.99% of the cats are

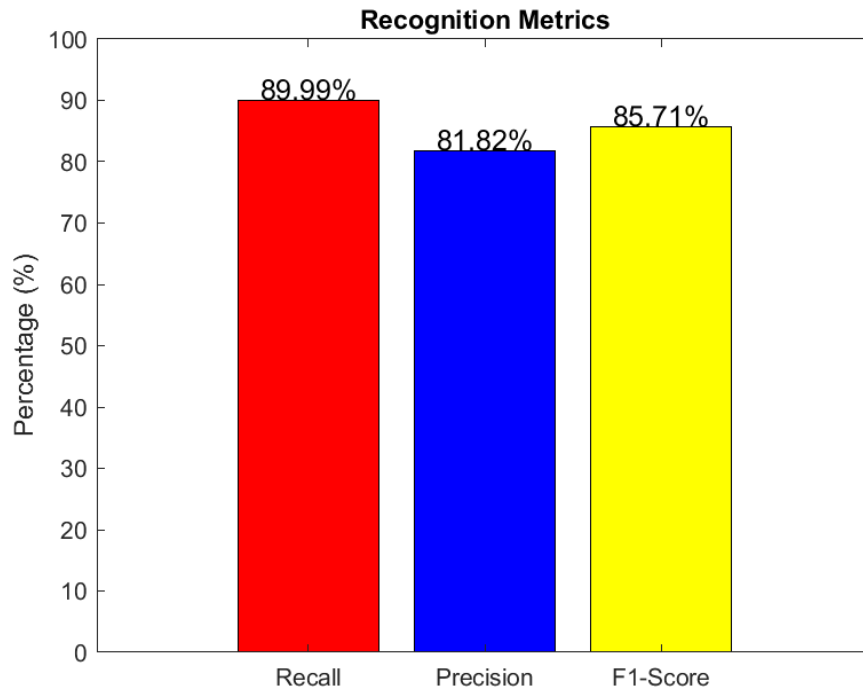


Figure 4.28: Recognition and Classification Metrics of the System.

successfully detected. In simpler words, it successfully detected 89.99% of the actual cats present. This shows that the system is highly sensitive and rarely misses a real threat. Besides, precision is used as another metrics to measure how many detected cases are actually cats. In other words, it measures the accuracy of the classification model. Equation 4.3 presents the formula for precision.

$$Precision = \frac{TP}{TP + FP} \quad (4.3)$$

where

$FP$  = False Positive

The value obtained for precision is 81.82% as shown in Figure 4.28 which indicates that out of all the detections made, 81.82% of them were correctly identified as cat. With this high value of precision, chances of false alarms can be greatly reduced. At the same time, accurate deterrent response can be assured. F1-score refers to the harmonic mean of recall and precision thus its value is calculated using the value of recall and precision as shown in Equation 4.4.

$$F1\text{-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4.4)$$

As a result, the value of F1-score obtained is 85.71%. This balanced score reflects the system's great ability and capability in detecting and classifying cats accurately. All in all, these results confirm that the Haar Cascade classifier performs reliably and is well-suited for real-time smart farming system.

#### 4.4.1.3 Detection Latency Analysis

Detection latency analysis is done by measuring the time taken for the system to detect cats across multiple trails to evaluate the system performance in detecting cats. Based on the diagram in Figure 4.29, it can be observed that the detection latency lies within the range of 0.7 s to 1.65 s. The detection latency includes the total time taken starting from the camera activating, scanning the live frame, system processing the input

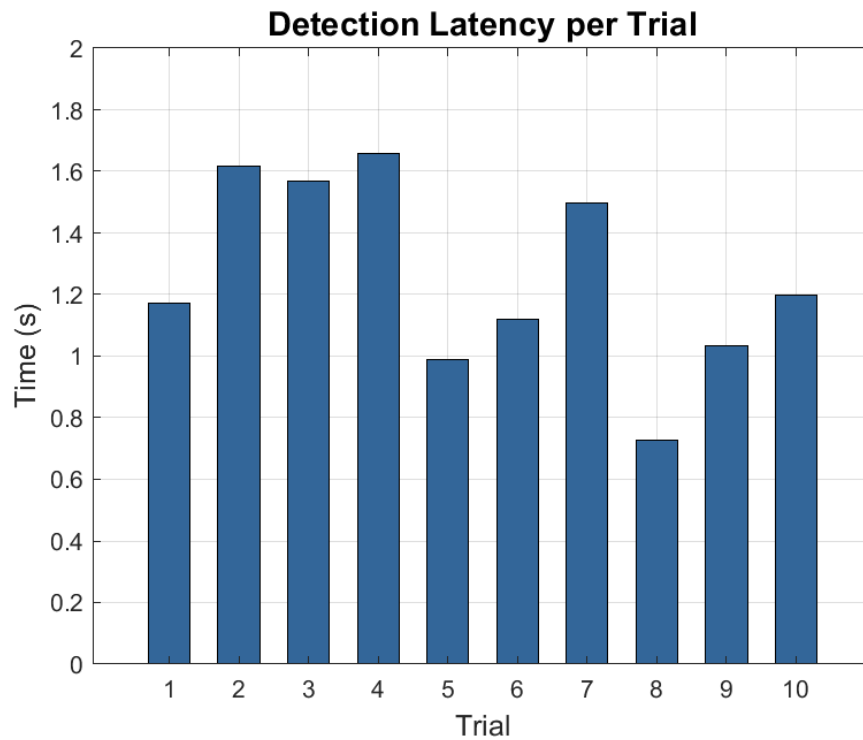


Figure 4.29: Detection Latency of Smart Farming System.

through the machine learning classifier, classification of the animal, specifically detecting cats and ending with the confirmation of cat presence. The results obtained based on Figure 4.29 proves that the system is quick in detecting cats. It is highly responsive to the presence of cat. Thus, it can be concluded that the system achieves high level of efficiency in detecting cat, contributing to a reliable operation for real-time smart farming system.

#### **4.4.2 Repelling Mechanism Analysis**

The performance of the repelling mechanism is tested in this analysis where analysis is carried out to determine the effectiveness of the system in repelling the cats. In other words, repelling mechanism analysis explains the repelling latency and the repelling success rate achieved by the system. This analysis is important to reflect the system's ability to fulfill the core objectives of this project, which is to repel the cats away in order to protect the crop field.

##### **4.4.2.1 Repelling Latency Analysis**

Repelling latency is defined as the time taken for the smart farming system to trigger the repelling mechanism and complete the repelling action. It includes the time taken to complete the final stage of the system operation process which is deterrence action module. By referring to Figure 4.30, the repelling latency of the system ranges from 0.2 s to 1.5 s. This result proves that the system is able to response rapidly upon detecting the presence of cats. In the majority of cases, the system manages to execute the repelling mechanism immediately by triggering the buzzer to sound on right after the system confirms the presence of cat. All in all, the system has successfully demonstrated its effectiveness in executing an immediate repelling response to the presence of cats.

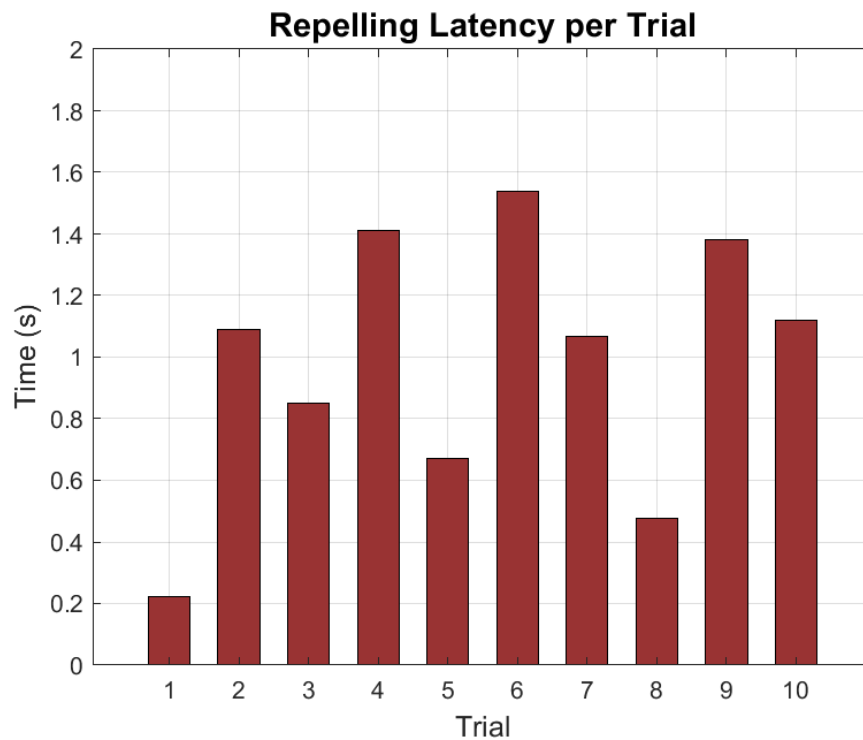


Figure 4.30: Repelling Latency of Smart Farming System.

#### 4.4.2.2 Repelling Success Rate

The repelling success rate of the smart farming system is obtained by evaluating the system performance, mainly focusing on the accuracy of the classification model and responsiveness of the repelling mechanism. These two aspects serve as the key factors that contribute significantly to the system's high repelling success rate. They work together to detect cats accurately and repel the cats immediately without any delay. At the same time, all of the processes should be performed rapidly. Hence, a high repelling success rate can be obtained if all these core aspects are achieved. Figure 4.31 shows that a repelling success rate of 80% to 100% is achieved by the system. By way of explanation, the reason behind the high repelling success rate is demonstrated in such a way that when a cat is present, it will be quickly and accurately detected by the system. After this, the buzzer will be activated immediately to emit a buzzing sound. As a result, the cat will fail to damage the crop since it is being driven away before it

manages to enter the crop field. Therefore, it can be said with certainty that the system emerged as a promising solution to protect the crops.

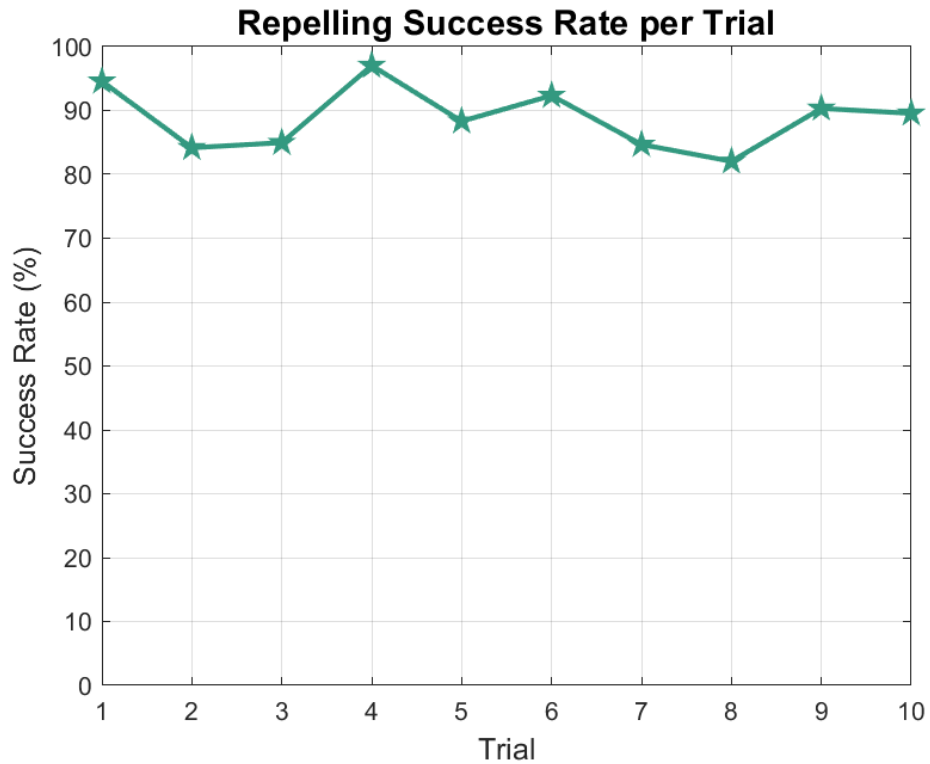


Figure 4.31: Repelling Success Rate of Smart Farming System.

#### 4.4.3 Total Response Time Analysis

The total response time of the smart farming system is another key aspect that has been analyzed using the logging event function introduced by the system. In general, the total response time represents the full operational cycle of the system. The response time includes duration for motion detection, camera activation, object classification, image acquisition, real-time telegram message delivery and buzzer activation. The total time is recorded from the beginning to the end of system operation. From the latency perspective as mentioned in the previous subsections, the total response time is the summation of detection latency and repelling latency. By referring to Figure 4.32, an average total response time that ranges from 4 s to 5 s is obtained through multiple test runs. This implies that it takes the smart farming system approximately 4.5 s to

detect and repel the cat away from the location where the system is placed at. This rapid response ensures that the repelling action is activated almost immediately after a cat is detected, minimizing the chance for crop damage. Besides, this rapid response also proves that the functional modules for smart farming system are integrated smoothly, leading to a successful development of an advanced system. All in all, this developed project signifies a holistic approach towards system reliability in real-time scenarios.

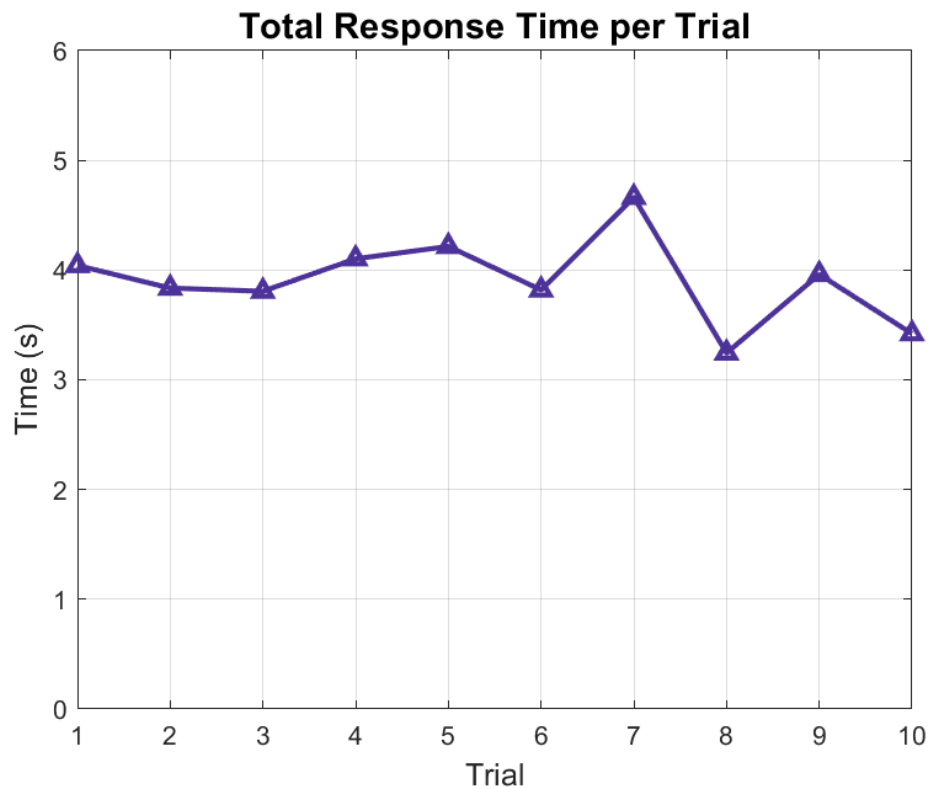


Figure 4.32: Total Response Time of Smart Farming System.

#### 4.4.4 Condition-based Accuracy Analysis

The performance of the system under various affecting factors is analyzed in this subsection. Factors such as cat species variation, lighting conditions, detection distance and number of cats are used to examine the accuracy and reliability of the system. Condition-based accuracy analysis is significant to ensure that the system



performs reliably and maintains robustness in real-time scenario that is full of uncertainty.

#### 4.4.4.1 Cat Species Variation

The performance of AI-driven IoT-based Smart Farming System is verified through testing on multiple species of cat. The diagrams presented in Figure 4.33 to Figure 4.35 present that the system is capable of identifying cats regardless of species difference. These results prove that the machine learning classifier which is known as Haar Cascade classifier is intelligent in classifying cats. It is trained to classify cats based on the features of cats. For this reason, Haar Cascade classifier has proven to be instrumental in animal classification, especially cats in this case. To further support this study, another case study involving a tiger is being carried out. This is because a tiger looks similar to a cat, therefore it might cause confusion to the classifier during the animal classification phase. However, there is no response from the system when



Figure 4.33: White Cat is Detected.



Figure 4.34: Gray Cat is Detected.



Figure 4.35: Orange Cat is Detected.

the system is tested with an image of a tiger. Furthermore, additional case studies involving dogs are also being conducted. The system shows no response to dogs. This result proves that the classifier performs excellently in detecting specific animals like cats. Hence, there is no doubt that this system is reliable since its performance is well-supported by different valid case studies.

#### 4.4.4.2 Lighting Conditions

To evaluate the performance of the smart farming system under different lighting conditions, another case study is investigated where the system is placed in a dark environment. Figure 4.36 shows that the system is able to detect and classify cats even though the surrounding environment has weak illumination. This is because the machine learning classification model is well-trained to perform classification with great tolerance to the illumination variations. With this comprehensive approach, this system achieves strong performance in detecting cats regardless of the lighting conditions. For this reason, this smart farming system is a stable and effective tool to protect the crops at day and night since it is immune to lighting conditions.

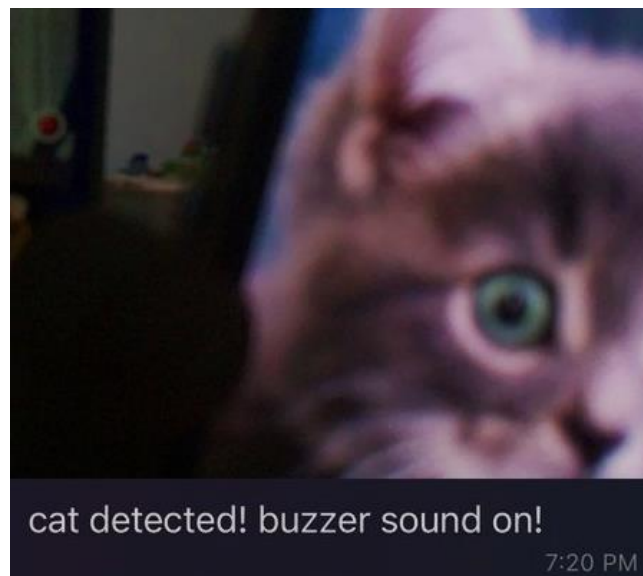


Figure 4.36: Cat is Detected in Dark Condition.

#### 4.4.4.3 Detection Distance

Detection distance is another factor taken into consideration when evaluating the performance of the smart farming system. It is important to ensure that the system can perform reliably under varied conditions regardless of the distance. The evaluation of



Figure 4.37: Short Detection Distance.

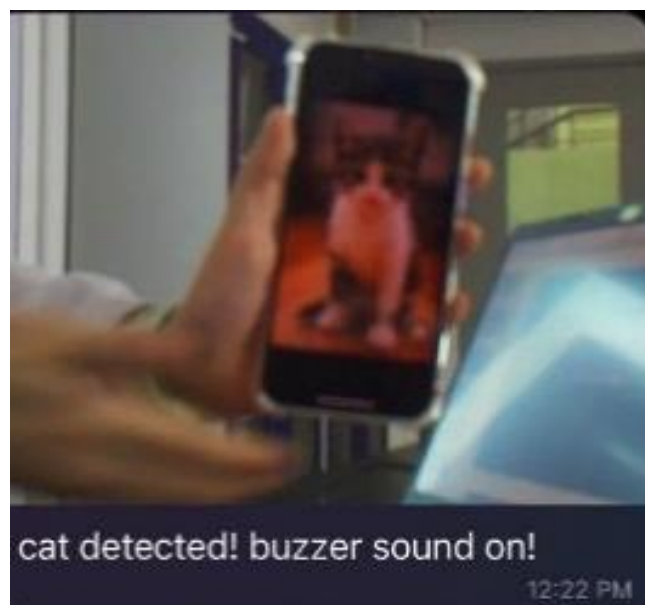


Figure 4.38: Long Detection Distance.

system performance at different distances is shown in Figure 4.37 and Figure 4.38, respectively. Although the cat is far away from the system and it is small in size, the system still manages to detect the cat. Therefore, this proves that the system is a powerful tool for wide area surveillance in agricultural settings.

#### 4.4.4.4 Number of Cats

By referring to Figure 4.39, multiple cats are detected by the system. This result shows that the system remains its accuracy to detect cats in complex scenes. In addition, the system introduces no latency in detecting multiple cats compared to detecting a single cat. In other words, the model can handle multiple cats in a single frame and perform rapid classification regardless of the number of objects in the frame. Therefore, it expands the system capability for scalability and accuracy in real-time scenarios.

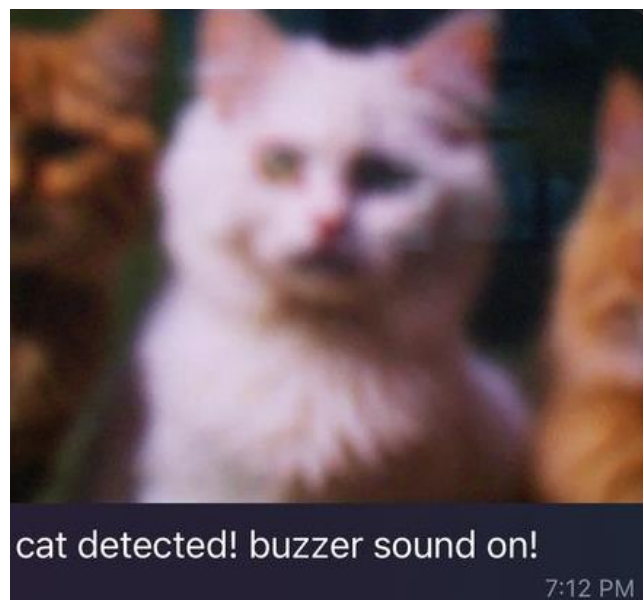


Figure 4.39: Multiple Cats are Detected.

## **CHAPTER 5**

### **CONCLUSIONS AND RECOMMENDATIONS**

#### **5.1 Conclusion**

In conclusion, a smart farming system powered by AI and IoT has been successfully developed and implemented to achieve the objectives of this project. The developed system is able to detect the presence of cats and execute immediate repellent action to the cats. At the same time, the system successfully interacts with the IoT cloud by delivering real-time data to user's Telegram via cloud-based communication. A software program is successfully developed using machine learning algorithms to perform accurate object classifications, specifically focusing on cats. Besides, the integration between multiple functional modules is successful where all functional components work together to contribute a seamless operation in detecting, classifying, and repelling animals efficiently and autonomously. This AI-driven IoT-based Smart Farming System has demonstrated a successful integration between intelligent animal classification with automated repelling mechanism. Data from the detection module is passed in real-time to the classification module, which identifies the species of the approaching animal. Based on the classification result, the system triggers the appropriate deterrent through the output control module and sends timely alert messages. All in all, the experimental results prove that this system is a powerful tool for surveillance of crop field as it not only fulfils the primary objective of this project but also shows great potential in enhancing the system's accuracy, efficiency and reliability in real-time scenarios. In addition, this system stands out for its remarkable performance in optimizing energy consumption.

## **5.2 Recommendation**

While this project has offered promising results, there are several areas that can be improved in future development. Firstly, a stepper motor could be introduced to the system to provide mobility. By integrating stepper motors into the system, the system is empowered to function like a surveillance robot, capable of autonomously patrolling the crop field. With this future improvement, the system will not remain stationary in that particular location, instead, it becomes an active device that moves around the crop field without any restrictions. Furthermore, the system's capability can be extended by developing more advanced AI models for real-time detection and classification for multiple classes of animals. On the other hand, it is recommended that solar panel charging could be introduced to the system, enabling sustainable operation with enhanced energy efficiency.

## REFERENCES

- Ada, L. and Dicola, T., n.d. PIR Motion Sensor. [online] Available at: <https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/overview> [Accessed 23 August 2024].
- Aloysius, N. and Geetha, M. (2017). A review on deep convolutional neural networks. 2017 International Conference on Communication and Signal Processing (ICCSP). <https://doi.org/10.1109/iccsp.2017.8286426>.
- BasuMallick, C., 2022. What Is Raspberry Pi? Models, Features, and Uses. [online] Available at: [https://www.spiceworks.com/tech/networking/articles/what-is-raspberry-pi/#\\_004](https://www.spiceworks.com/tech/networking/articles/what-is-raspberry-pi/#_004) [Accessed 22 August 2024].
- Corporate Finance Institute Team, 2023. Python (in Machine Learning). [online] Available at: <https://corporatefinanceinstitute.com/resources/data-science/python-in-machine-learning/#:~:text=Python%20includes%20a%20modular%20machine,the%20Python%20frameworks%20and%20libraries.> [Accessed 23 August 2024].
- Feng, J. and Xiao, X., 2022. Multiobject Tracking of Wildlife in Videos Using Few-Shot Learning. *Animals*, 12(9), p.1223. doi:<https://doi.org/10.3390/ani12091223>.
- Global Sensor Technology, 2021. *Active Infrared Vs. Passive Infrared*. [online] Available at: <https://www.gst-ir.net/news-events/infrared-knowledge/491.html#:~:text=In%20summary%2C%20the%20main%20difference,objects%20in%20their%20surrounding%20environment.> [Accessed 23 August 2024].
- Keita, Z., 2022. *YOLO Object Detection Explained*. [online] Available at: <https://www.datacamp.com/blog/yolo-object-detection-explained> [Accessed 10 August 2024].
- Kellenberger, B., Volpi, M., and Tuia, D. "Fast animal detection in UAV images using convolutional neural networks," 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Fort Worth, TX, USA, 2017, pp. 866-869, doi: 10.1109/IGARSS.2017.8127090.
- Konkala Venkateswarlu Reddy, Reddy, K., Veerapu Goutham, Mahesh, M., Nisha, J.S., Gopinath Palanisamy, Golla, M., Swetha Purushothaman, Katangure Rithisha Reddy and Ramkumar, V. (2024). Edge AI in sustainable farming: Deep Learning-



- Driven IoT Framework to Safeguard Crops from Wildlife Threats. *IEEE Access*, 12, pp.77707–77723. doi: <https://doi.org/10.1109/access.2024.3406585>
- Kundu, R., 2022. *Everything you need to know about Few-Shot Learning*. [online] Available at: <<https://blog.paperspace.com/few-shot-learning/>> [Accessed 7 August 2024].
- Lekhaa, T. R., Sumathi, Dr. P., Rajan, B. S., Vikas, N. S., Murugan, S. A., and Kumar, D. S., 2023. Airep: Ai And Iot Based Animal Recognition And Repelling System For Smart Farming. *NVEO - NATURAL VOLATILES & ESSENTIAL OILS Journal / NVEO*, [online] pp.1873–1883. Available at: <<https://www.nveo.org/index.php/journal/article/view/4959>> [Accessed 25 Aug. 2024].
- Logos-world, 2024. *Telegram Logo*. [online] Available at: < <https://logos-world.net/telegram-logo/>> [Accessed 25 August 2024].
- Mishra, A. and Yadav, K.K., 2024b 'Smart Animal Repelling Device: Utilizing IoT and AI for Effective Anti-Adaptive Harmful Animal deterrence,' *BIO Web of Conferences*, 82, p. 05014. <https://doi.org/10.1051/bioconf/20248205014>.
- Raspberry Pi, 2019. *Raspberry Pi 4 Computer Model B*. [online] Available at: <<https://docs.rs-online.com/b1fb/0900766b816fa153.pdf>> [Accessed 22 August 2024].
- Raspberry Pi, n.d. *Raspberry Pi Camera Module 2*. [online] Available at: <<https://www.raspberrypi.com/products/camera-module-v2/>> [Accessed 23 August 2024].
- Ra20Ga, 2021. *Python programming logo on transparent background PNG*. [online] Available at: <<https://similarpng.com/python-programming-logo-on-transparent-background-png/>> [Accessed 23 August 2024].
- Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. *You Only Look Once: Unified, Real-Time Object Detection*. [online] Available at: <[https://www.cv-foundation.org/openaccess/content\\_cvpr\\_2016/papers/Redmon\\_You\\_Only\\_Look\\_CVPR\\_2016\\_paper.pdf](https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf)> [Accessed 8 August 2024].
- Rubi, J., V, S., Dhivya, A. Josephin Arockia and Vijayalakshmi, A., 2024. A Review of Crop Protection Methods in Agricultural Fields. *IRO Journal on Sustainable Wireless Systems*, [online] 6(1), pp.75–83. Available at: <<https://irojournals.com/irosws/article/view/6/1/6>> [Accessed 23 Aug. 2024].
- Sayagavi, A.V., Sudarshan, T.S.B., Ravoor, P.C., 2021. Deep Learning Methods for Animal Recognition and Tracking to Detect Intrusions. In: Senjyu, T., Mahalle, P.N., Perumal, T., Joshi, A. (eds) *Information and Communication Technology for Intelligent Systems. ICTIS 2020. Smart Innovation, Systems and Technologies*, vol 196. Springer, Singapore. [https://doi.org/10.1007/978-981-15-7062-9\\_62](https://doi.org/10.1007/978-981-15-7062-9_62)

Thailappan, D., 2024. *An Introduction to Few-Shot Learning*. [online] Available at: <<https://www.analyticsvidhya.com/blog/2021/05/an-introduction-to-few-shot-learning/>> [Accessed 7 August 2024].

Uptodown, n.d. *VNC viewer* [online]. Available at: <<https://vnc-viewer.en.uptodown.com/android>> [Accessed 31 March 2025].

Wangchuk, S., Bond, J., Thwaites, R. and Finlayson, M., 2023. Exploring Human–Wildlife Conflict and Implications for Food Self-Sufficiency in Bhutan. *Sustainability*, [e-journal] 15(5), p.4175. <https://doi.org/10.3390/su15054175>.

## APPENDICES

### Appendix A: Python Program for Integrated Smart Farming System.

```
#!/usr/bin/python3
import cv2
import os
import time
import datetime
import requests
import RPi.GPIO as GPIO
import signal
from picamera2 import Picamera2

# Telegram configuration
BOT_TOKEN = "8115057421:AAHYv-nEQ7nnLSFs-iWxG-
uUChD3DgbtGwg" #private bot token for cropguard
CHAT_ID = "983344994" #private chat ID for cropguard

# PIR sensor and buzzer GPIO setup
PIR_PIN = 4 #GPIO pin connected to PIR sensor
BUZZER_PIN = 17 #GPIO pin connected to buzzer

GPIO.setmode(GPIO.BCM)
GPIO.setup(PIR_PIN, GPIO.IN) #PIR sensor is configured as input
GPIO.setup(BUZZER_PIN, GPIO.OUT) #Buzzer is configured as output
GPIO.output(BUZZER_PIN, GPIO.LOW) #Ensure buzzer is off initially
```

```

# Log file configuration for event tracking
log_file = "detection_log_file.txt"
def log_event(message):
    timestamp
    =
datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    with open(log_file, "a") as f:
        f.write(f"[{timestamp}] {message}\n")

def handle_exit(signum, frame):
    print("\nProgram interrupted. Exiting gracefully...")
    log_event("Program exited by user")
    GPIO.cleanup()
    picam2.close()
    cv2.destroyAllWindows()
    exit(0)

signal.signal(signal.SIGINT, handle_exit)

# Function to send a Telegram message with image
def send_telegram_message_with_image(message, image_path):
    url = f"https://api.telegram.org/bot{BOT_TOKEN}/sendPhoto"
    data = {"chat_id": CHAT_ID, "caption": message}
    with open(image_path, 'rb') as photo:
        files = {"photo": photo}
    try:
        response = requests.post(url, data=data, files=files)
        if response.status_code != 200:
            print("Failed to send the photo.")
            log_event("Failed to send Telegram image")
    except Exception as e:
        print(f"Failed to send Telegram message with image: {e}")
        log_event(f"Telegram error: {e}")

```

```

# Load Haar Cascade classifier to detect cats
cat_detector = cv2.CascadeClassifier("/home/pi/cat_detection/haarcascade_frontalcatface.xml")

# Initialize Picamera2
picam2 = Picamera2()
picam2.configure(picam2.create_still_configuration(main={"format":
'RGB888', "size": (640, 480)})) #Enlarged resolution

# Create directory for detected cat faces
output_directory = "detected_cats"
os.makedirs(output_directory, exist_ok=True)

last_detection_time = 0 #Tracks the time of last detection
cooldown = 10 #Cooldown period in seconds
min_active_time = 10 #Camera stays active for at least 10 s

# Allow PIR sensor to stabilize for 30s
print("Initializing PIR sensor... Waiting for stabilization (30s)")
log_event("Initializing PIR sensor  Waiting 30 seconds")
time.sleep(30)
print("PIR sensor ready.")
log_event("PIR sensor ready")

print("Waiting for motion... (Press Ctrl+C to exit)")
log_event("System started Waiting for motion")

try:
    while True:
        if GPIO.input(PIR_PIN): # Motion detected
            print("Motion detected! Confirming...")
            log_event("Motion detected")
            time.sleep(0.5) #Small delay to prevent false triggers

```

```

        if GPIO.input(PIR_PIN): # Double-check motion
            print("Confirmed! Activating camera...")
            picam2.start()
            time.sleep(2) #Allow adjustment of camera
            motion_start_time = time.time()

        cat_detected = False #Flag to track if a cat was detected

        while time.time() - motion_start_time < min_active_time:
            im = picam2.capture_array()
            grey = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
            cats = cat_detector.detectMultiScale(grey, 1.1, 5)

            if len(cats) > 0 and (time.time() - last_detection_time >
cooldown):
                print("Cat detected! Capturing image and activating
buzzer...")

                timestamp =
datetime.datetime.now().strftime("%m-%d_%H-%M")
                filename = os.path.join(output_directory,
f"cat_{timestamp}.jpg")
                cv2.imwrite(filename, im)
                log_event(f"Cat detected - Image saved as {filename}")

                # Activate buzzer only when a cat is detected
                GPIO.output(BUZZER_PIN, GPIO.HIGH)
                log_event("Buzzer activated")

                send_telegram_message_with_image("cat detected! buzzer
sound on!", filename)

                last_detection_time = time.time()
                cat_detected = True

```

```
        for (x, y, w, h) in cats:
            cv2.rectangle(im, (x, y), (x + w, y + h), (255, 0, 0), 2)

        cv2.imshow("Cat Detection", im)
        cv2.waitKey(1)

    print("Camera deactivating...")
    log_event("Camera deactivated")
    picam2.stop()

    # Turn off buzzer only when no motion is detected
    print("Waiting for no motion to turn off buzzer...")
    while GPIO.input(PIR_PIN):
        time.sleep(0.5)

    GPIO.output(BUZZER_PIN, GPIO.LOW) #Deactivate buzzer
    print("Buzzer turned off.")
    log_event("Buzzer deactivated No motion detected")

except KeyboardInterrupt:
    handle_exit(None, None)
```

## Appendix B: Program Code to Create Vector File of Positive Samples for Training Automation in Bash.

```
#!/bin/sh

vec=binary_description
info=positive_description.txt
bg=negative_description.txt

# Uncomment the next 4 variables for LBP training.
#featureType=LBP
#data=lbp_cascade_frontalcatface/
#dst=../cascades/lbp_cascade_frontalcatface.xml
#mode=BASIC

# Uncomment the next 4 variables for Haar training with basic
# features.
featureType=HAAR
data=haar_cascade_frontalcatface/
dst=../cascades/haar_cascade_frontalcatface.xml
mode=BASIC

# Uncomment the next 4 variables for Haar training with
# extended features.
#featureType=HAAR
#data=haar_cascade_frontalcatface_extended/
#dst=../cascades/haar_cascade_frontalcatface_extended.xml
#mode=ALL

# Set numPosTotal to be the line count of info.
numPosTotal=`wc -l < $info`

# Set numNegTotal to be the line count of bg.
numNegTotal=`wc -l < $bg`
```



```
numPosPerStage=$((($numPosTotal*9/10))
numNegPerStage=$((($numNegTotal*9/10))
numStages=20
minHitRate=0.995
maxFalseAlarmRate=0.5

# Ensure that the data directory exists and is empty.
if [ ! -d "$data" ]; then
    mkdir "$data"
else
    rm "$data/*.xml"
fi

opencv_createsamples -vec "$vec" -info "$info" -bg "$bg" \
    -num "$numPosTotal"
opencv_traincascade -data "$data" -vec "$vec" -bg "$bg" \
    -numPos "$numPosPerStage" -numNeg "$numNegPerStage" \
    -numStages "$numStages" -minHitRate "$minHitRate" \
    -maxFalseAlarmRate "$maxFalseAlarmRate" \
    -featureType "$featureType" -mode "$mode"

cp "$data/cascade.xml" "$dst"
```

## Appendix C: Program Code to Generate Annotated Training Data for the Classification Model.

```
import cv2
import glob
import math
import sys

outputImageExtension = '.out.jpg'

def equalizedGray(image):
    return cv2.equalizeHist(cv2.cvtColor(
        image, cv2.COLOR_BGR2GRAY))

def describeNegativeHelper(imagePath, output):
    outputImagePath = '%s%s' % (imagePath, outputImageExtension)
    image = cv2.imread(imagePath)
    # Save an equalized version of the image.
    cv2.imwrite(outputImagePath, equalizedGray(image))
    # Append the equalized image to the negative description.
    print >> output, outputImagePath

def describeNegative():
    output = open('negative_description.txt', 'w')
    # Append all images from Caltech Faces 1999, since all are
    # non-cats.
    for imagePath in glob.glob('faces/*.jpg'):
        if imagePath.endswith(outputImageExtension):
            # This file is equalized, saved on a previous run.
            # Skip it.
            continue
        describeNegativeHelper(imagePath, output)
    # Append all images from the Urtho negative training set,
    # since all are non-cats.
```

```

for imagePath in glob.glob('urtho_negatives/*.jpg'):
    if imagePath.endswith(outputImageExtension):
        # This file is equalized, saved on a previous run.
        # Skip it.
        continue
    describeNegativeHelper(imagePath, output)
# Append non-cat images from VOC2007.
input = open('VOC2007/ImageSets/Main/cat_test.txt', 'r')
while True:
    line = input.readline().rstrip()
    if not line:
        break
    imageNumber, flag = line.split()
    if int(flag) < 0:
        # There is no cat in this image.
        imagePath = 'VOC2007/JPEGImages/%s.jpg' % imageNumber
        describeNegativeHelper(imagePath, output)

def rotateCoords(coords, center, angleRadians):
    # Positive y is down so reverse the angle, too.
    angleRadians = -angleRadians
    xs, ys = coords[:,2], coords[1::2]
    newCoords = []
    n = min(len(xs), len(ys))
    i = 0
    centerX = center[0]
    centerY = center[1]
    cosAngle = math.cos(angleRadians)
    sinAngle = math.sin(angleRadians)
    while i < n:
        xOffset = xs[i] - centerX
        yOffset = ys[i] - centerY
        newX = xOffset * cosAngle - yOffset * sinAngle + centerX

```



```

# Straighten the coordinates of the features.
newCoords = rotateCoords(
    coords, eyesCenter, eyesAngleRadians)

# Make the face as wide as the space between the ear bases.
# (The ear base positions are specified in the reference
# coordinates.)
w = abs(newCoords[16] - newCoords[6])
# Make the face square.
h = w
# Put the center point between the eyes at (0.5, 0.4) in
# proportion to the entire face.
minX = eyesCenter[0] - w/2
if minX < 0:
    w += minX
    minX = 0
minY = eyesCenter[1] - h*2/5
if minY < 0:
    h += minY
    minY = 0

# Crop the face.
crop = straight[minY:minY+h, minX:minX+w]
# Convert the crop to equalized grayscale.
crop = equalizedGray(crop)
# Return the crop.
return crop

def describePositive():
    output = open('positive_description.txt', 'w')
    dirs = ['CAT_DATASET_01/CAT_00',
            'CAT_DATASET_01/CAT_01',

```

```
'CAT_DATASET_01/CAT_02',
'CAT_DATASET_02/CAT_03',
'CAT_DATASET_02/CAT_04',
'CAT_DATASET_02/CAT_05',
'CAT_DATASET_02/CAT_06']
```

```
for dir in dirs:
```

```
    for imagePath in glob.glob('%s/*.*jpg' % dir):
        if imagePath.endswith(outputImageExtension):
            # This file is a crop, saved on a previous run.
            # Skip it.
            continue

        # Open the '.cat' annotation file associated with this
        # image.
        input = open('%s.cat' % imagePath, 'r')

        # Read the coordinates of the cat features from the
        # file. Discard the first number, which is the number
        # of features.
        coords = [int(i) for i in input.readline().split()[1:]]

        # Read the image.
        image = cv2.imread(imagePath)

        # Straighten and crop the cat face.
        crop = preprocessCatFace(coords, image)

        if crop is None:
            print >> sys.stderr, \
                'Failed to preprocess image at %s.' % \
                imagePath
            continue

        # Save the crop.
        cropPath = '%s%s' % (imagePath, outputImageExtension)
        cv2.imwrite(cropPath, crop)

        # Append the cropped face and its bounds to the
        # positive description.
        h, w = crop.shape[:2]
```

```
print >> output, cropPath, 1, 0, 0, w, h

def main():
    describeNegative()
    describePositive()

if __name__ == '__main__':
    main()
```