ASHESI UNIVERSITY

DESIGN AND IMPLEMENTATION OF A TOMATO LEAF BLIGHT DISEASE DETECTION SYSTEM USING MACHINE LEARNING AND IMAGE PROCESSING.

CAPSTONE PROJECT

B.Sc. Computer Engineering

Michael Kwame Atansa Ansah

2021

ASHESI UNIVERSITY

DESIGN AND IMPLEMENTATION OF A TOMATO LEAF BLIGHT DISEASE DETECTION SYSTEM USING MACHINE LEARNING AND IMAGE PROCESSING.

CAPSTONE PROJECT

Capstone Project submitted to the Department of Engineering, Ashesi University in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Engineering.

Michael Kwame Atansa Ansah

2021

DECLARATION

I hereby declare that this capstone is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature: ……**M.K.A.A**……

Candidate's Name:……Michael Kwame Atansa Ansah……

Date: ……**April 23, 2021**……

I hereby declare that the preparation and presentation of this capstone were supervised in accordance with the guidelines on the supervision of capstone laid down by Ashesi University.

Supervisor's Signature: ……

Supervisor's Name: ……

Date: ……

Acknowledgments

To my supervisor, Elena Rosca, whose encouragement and academic advice helped me undertake this project.

To my Father and mentor, Mr. Patrick Ansah, whose guidance and encouragement molded this project idea and brought it to successful completion.

Abstract

Tomato production in Sub-Saharan Africa can be plagued by a fungal disease known as leaf blight. The tomato blight disease can have devastating effects on smallholder farmers if measures are not taken early. Currently, the most common method of leaf disease diagnosis is an estimation by human judgment. This method is very error-prone and can be quite tedious given the volume of plants on a typical tomato farm. This research investigates the creation of a faster disease detection method using image processing and machine learning. Similar solutions in the past that employ these technologies typically have to connect to an offsite server to perform all processing and diagnosis. This project explores the deployment of compressed, quantized models on small edge devices which can produce accurate results at a significantly low cost. The tomato blight disease is characterized by brown leaf spots. The system could classify a diseased leaf with brown spots from a healthy leaf with an accuracy of 90% ± 12.32.

Table of Contents

List of Tables

Table Of Figures

Chapter 1: Introduction

Commented [ER1]: Please use the document formatting.

Commented [MA2R1]:

1.1 Background

Africa has immense untapped agricultural potential as it holds 60% of the world's farmable land (Islam & Digneffe, 2012). Agriculture in Africa is predominantly carried out by smallholder farmers (Martey, Al-Hassan, & Kuwornu, 2012). Smallholder farmers cultivate subsistence crops on small-based plots and rely almost exclusively on family labor. These smallholder farmers have very few decision support systems to inform their choices on the farm. Many variables like crop yield, topography, and organic matter content nutrient levels can provide insights that may guide farmers' decisions. Unfortunately, subsistence farming relies mainly on household decision-making and human discretion in the cultivation process(Martey,2012). For Ghanaian farmers to participate in commercialized, large-scale agriculture, they will require better-informed decision-making on farms.

Commented [ER3]: I am not sure that I really understand the point you are trying to make here.

Commented [ER4]: Same here is not clear.

Farmers can make better management choices through the use of precision agriculture. Precision agriculture is a farming management concept based on monitoring, measuring, and responding to variability in crops (Mavridou, Vrochidou, Papakostas, Pachidis, & Kaburlasos, 2019). Implementing precision agriculture creates a decision support system for farm management by optimizing the returns on input while preserving resources. It involves using technology to extract information from the crops and soil to maximize the health & yield of the crops and the productivity of the farmers. In today's world of technological advancement, data concerning the farm is garnered using a vast array of sensors that measure numerous variables

from plant moisture content to chlorophyll levels. (Reyns, Missotten, Ramon, & De Baerdemaeker, 2002)

However, precision agriculture can be quite expensive for smallholder farmers to implement because it requires expensive technology. This technology has been used to support precision agriculture through automated approaches to tasks that are conventionally performed manually. The viability of precision agriculture was speculated to be inversely proportional to the cost of the technologies and their ease of use (Silva, Pinto, Müller, & Moura, 2007). According to Goldman Sachs's research on the internet of things, the cost of IoT sensors has been steadily dropping since 2004 (Jankowski, Covello, Bellini, Ritchie, & Costa, 2014).

1.2 Motivation

Tomato is mostly produced in seven out of the ten regions in Ghana. (Melomey, et al., 2019) Tomato production in Ghana is seasonal. During the rainy season, tomato is produced in abundance, whereas in the dry season, it gets scarce because of the dry weather conditions. Demand for tomatoes during the dry season is typically larger than the supply of tomatoes. For this reason, greenhouses have become more common in Ghana. Greenhouses give farmers longer growing seasons and more control over the environment in which their crops are cultivated. (Pallavi, Mallapur, & Bendigeri, 2017). There have been significant investments made in greenhouse agriculture in Ghana. The world Bank spent over $2,000,000.00 in funding the West African Agricultural Productivity Program to establish over 200 greenhouses in Ghana (Sarfo, 2018). If the microclimatic conditions in greenhouses are not controlled properly, fungal diseases can cause huge losses in such an intensive environment. (Gullino, Albajes, & Nicot, 2020). Tomato production in Sub-Saharan Africa can be plagued by fungal diseases such as Early

Blight, Late Blight, and Fruit Rot caused by *Phytophthora infestans, Alteranaria solani,* and *Phytophthora parasitica,* respectively (Dube, G., & Maphosa, 2020) . The late blight of potatoes was responsible for the Irish potato famine of the late 1840s. (Mizubuti & Fry, 2006).The scope of this project focuses on detecting leaf diseases affecting tomato production in Ghana. The detection method should use physical characteristics like lesions, blight, galls, rots, and wilts to non-intrusively discern whether a plant is diseased or not.

Current sensors on farms and greenhouses capture data and transport them to a remote location for processing. Such a system can be quite expensive to acquire, deploy and maintain, especially for Ghanaian subsistence farmers.

This project seeks to leverage on development of low-power microcontrollers that can run quantized versions of the complex machine learning algorithms locally. These embedded boards do not require access to the internet. They are essentially lower-cost hardware that can accomplish results similar to that of high-end hardware. They will ultimately provide the desired social impact at a fraction of the cost.

1.3 Problem Definition

Subsistence farmers in Ghana need a faster, more accurate, and less labor-intensive way of detecting blight on tomato plants. Tomato Blight has devastating effects on farms when it is not detected immediately. Hence, a reliable method of disease detection is imperative.

1.4 Proposed Solution

The main objective is to develop a low-cost farm surveillance device that detects tomato blight without connecting to a network beyond the farm's premises. The project is to be implemented on a single-acre farm. Should it be successful, it would be scaled up to be implemented on larger farms.

Commented [ER5]: Dear Michael. This is a great detailed plan. I think at the first look this should give a bit more insight in what you think the impact of the product would be. These details that you have here will fit very well in the beginning of chapter 3.

Chapter 2: Literature Review

Tomatoes are affected by a myriad of diseases in Ghana and Africa as a whole. However, there is no present locally automated real-time detection mechanism for these diseases in Ghana. Farmers have to rely on human discernment. A survey conducted on farmer's knowledge of how greenhouse diseases are spread revealed that roughly 64% of greenhouse farmers are unaware of how infectious diseases on tomatoes are spread (Sarfo, 2018). Hence, farmers require an inexpensive and accurate means of detecting tomato plant diseases.

Over the past few decades, technological strides have been made in efficient and accurate disease detection in agriculture. Some of which involve phenotyping and image processing. Mahlein compares image processing and phenotyping as disease detection methods. (Mahlein, 2016). He purports that a common downside of both ways is the large amount of data required for the effectiveness of both systems. Mahlein claims that data used in both detection methods must have data on the early disease symptoms and quantify the disease severity and healthy conditions. (Mahlein, 2016). Since both ways are heavily data reliant, a reliable data mining method is necessary for the system. The papers in the review discuss various disease detection methods. The most reoccurring methods mentioned are estimation by human judgment, microscopic evaluation of morphology, and microbiological diagnostic techniques. Another common denominator in the articles reviewed is the unreliability of human diagnosis. They are described as 'time-consuming,' 'subject to human bias' and require and 'experienced individuals with well-developed skills in diagnosis.'

All of the studies mention the systematic selection of relevant features from the RGB images as a potent disease detection method. (Behmann, Mahlein, Rumpf, Römer, & Plümer, 2015).Multiple papers in the study employ histograms that convey the color distribution of images to detect diseases. Another standard detection method is thresholding which involves classifying pixel groups within a specific color range as 'light' or 'dark' (Bradley & Roth, 2007). Thresholding is typically used to mask the diseased object of interest from the background or mask the diseased features from healthy features.

Difficulty in detection and low levels of accuracy are often the results of heterogenic conditions and low image quality. The most important principle for success is a sound standardized imaging procedure that yields repeatable results. In 1936, Riker and Riker emphasized the difficulties in diagnosing and detecting plant diseases. They gave an overview of the strengths and limitations of existing methods. They concluded that: plant protection and ecology of the future require innovative techniques with guaranteed higher accuracy because of human oversight. They assert that automated methods with reliability and accuracy are a necessary upgrade from the current visual estimation techniques. (Mahlein, 2016)

New disease detection methods which are more accurate with a lower risk of the error have been developed. These automated detection methods will replace unreliable visual inspection methods, particularly when trained experts on farms in Ghana are rare.

One critical technology used is a Convolutional Neural Network(CNN). CNNs are more efficient than previous plant disease diagnosis methods like support vector machines and K-means clustering. (Xie, et al., 2020). CNNs have their inherent limitations in disease detection,

however. CNNs are susceptible to environmental factors such as lighting and shielding by other leaves. CNNs can also have downfalls when the dataset is not adequately prepared. The image dataset used to train the CNN is subjected to digital processing to avoid overfitting and underfitting. Overfitting occurs when a CNN learns a function in a manner that is tightly fit to a limited set of data points. This digital processing may involve data augmentation, a process through which multiple nuanced duplicates of the image data populate the dataset (Shorten & Khoshgoftaar, 2019) . These duplicates typically have geometric transformations such as shearing, rotating, flipping, and scaling.

A significant advantage of microcontrollers (MCUs) with machine learning capability is developing applications that do not need processing power from the cloud. Processing through the cloud consumes a lot of energy, and it stands the threat of security breaches. These systems are described as 'smart and frugal' on countless occasions, highlighting their efficient use of available resources. However, their power constraints will require a reduction in the size of the data with which they work. (Martinez, Monton, Vilajosana, & Prades, 2015)Nevertheless, given the power constraints of MCUs, reducing data transmissions is crucial for their sustainability as this type of operation is more power demanding than computational tasks.

The most well-known Machine learning libraries that are ported onto MCUs are Scikit-Learn, TensorFlow, and PyTorch. (Sanchez-Iborra & Skarmeta, 2020)The papers report that all the frameworks for machine learning on MCUs focus on only a single machine learning algorithm, Neural Networks. However, independent developers and researchers have embarked on projects that employ ML techniques such as Naive Bayes classifier, Support Vector Machine, Relevance vector machine, decision trees, k-Nearest Neighbors (k-NN) on

microcontrollers. (Sanchez-Iborra & Skarmeta, 2020) Sanchez-Iborra claims that trees and random forest algorithms present the best performance for accuracy, classification, and speed. The next runner-up is a neural network.

Many Tech Behemoths are in support of the proliferation of the TinyML movement. A prime example is Google. Google have created an open-source development library known as TensorFlow. TensorFlow is a machine learning system that operates in both small-scale and large-scale environments. (Abadi, et al., 2016). TensorFlowlite is a version of the TensorFlow library that can create quantized versions of the TensorFlow models. These Quantized models are suitable enough to be run on resource-constrained embedded devices. (Louis, et al., 2019). TensorFlow Lite has two main sub-components an interpreter and a converter. This quantized pair of an interpreter and converter offers the following benefits: Lower latency for model predictions; Lower bandwidth consumption; Privacy for on-device data, Lower power consumption, and the ability to run without a constant internet connection. (Sanchez-Iborra & Skarmeta, 2020)

Chapter 3: Design

3.1 Requirements Specifications

3.1.1 Product perspective

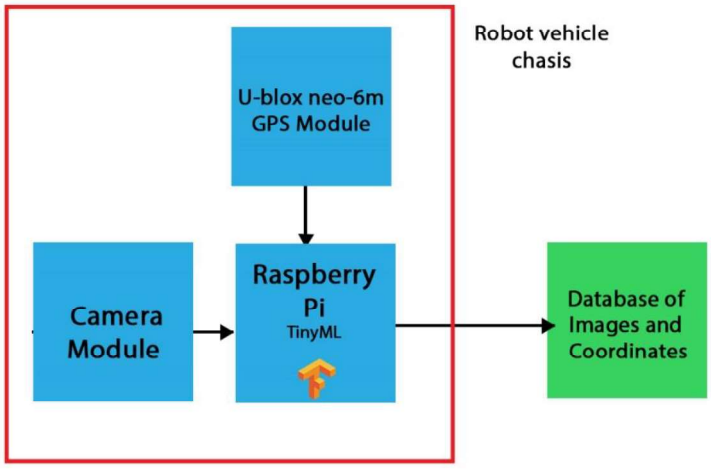

Figure 1: Block diagram of the disease detection system

Like most IoT projects, this farm disease detection system is made of hardware and software. The hardware consists of a camera interfaced with a raspberry pi, a GPS module, and a robot vehicle chassis, as shown in Figure 1. The system's hardware will be housed in the robot vehicle chassis. The software is made up of a front-end web app and a backend database. The raspberry pi creates the programming environment for image processing and machine learning and will run the disease detection model using Python and TensorFlow lite. The robot vehicle will convey the camera and a raspberry pi around the farm. The raspberry pi will communicate wirelessly with a database.

The UBlox Neo-6M GPS will be interfaced with the raspberry pi to retrieve the GPS coordinates of the diseased plant at a low cost. (Satria, Yana, Munadi, & Syahreza, 2017). The selected camera is the raspberry pi rev 1.3 camera. The camera has a high sensitivity, low noise, and crosstalk. It can operate within the temperature ranges of -30° to 70° making it versatile for deployment in various farm climates. The camera also possesses automatic exposure control functions, which are very necessary given that the image sensor has a limited dynamic range. A camera with automatic exposure control adapts to different light intensities (Murakami & Honda, 1996).

The system will be active during the day. During the nighttime, there is very little light available for the sensor of the camera. Such low-light conditions would yield inaccurate results since cost-effective components are in use in this system.

This system will be deployed in both greenhouses and outdoor farms which cultivate tomatoes in rows. Greenhouses are controlled environments with level ground and demarcated rows. Such an environment will be the most ideal for this system. Outdoor farms tend to be more irregular. Planting may not be done systematically, and the uneven soil path could quickly derail the robotic vehicle. The ideal farm for this system is a Greenhouse followed by an outdoor farm with plants cultivated in rows.

3.1.2 User interfaces

The user will be performing two main tasks using this system: Robot navigation and Disease monitoring

Robot navigation.

Robot navigation entails the user controlling the robot's travel path and the positioning of the camera. The user has two ways to go about this. The first way is to choose a predetermined path for the robot to travel and position its camera. This fixed surveillance routine will be carried out by the system at the press of the button. As mentioned earlier, Tomato farms and greenhouses typically have crops planted in rows. The robot will traverse the paths between crop rows with its camera in different positions each time to survey the entirety of the farm. The second way of navigating the robot will be with a manual Bluetooth remote control. With this method, the user controls the navigation of the robot and the camera with an android application. The android control of the application is more of a secondary method made available for the rare occasion where the user would want to control the robot himself.

Disease Monitoring.

This involves the user overseeing all the data the system produces through a web app. This supervisory activity is the staple of precision agriculture in this project. The user can discern the extent of the disease spread on the farm with images and statistics. The farmer will make better, informed decisions about managing the farm using these statistics. The entire project boils down to this critical end goal: Making better decisions on the farm with more accurate data.

Image data can be sensitive. Hence, users can access data on the web app only with administrator access. An account and login mechanism enforces administrator access. The account system grants only authorized personnel access to farm image data. A dashboard precedes the login page. The web app dashboard creates notifications every time a diseased

plant is detected and gives an overview of the number of positive tomato blight cases on the farm and the percentage damage suffered by the leaves.

3.1.3 Hardware interfaces

The system has various modules and components. Each of them transmits and receives data and power using distinctive hardware interfaces.

a) CSI

As shown in Figure 2, the raspberry pi Camera Rev 1.3 module will be interfaced with the raspberry pi to capture input images. The camera uses the CSI interface to communicate with the pi. The CSI bus enables high data transfer rates, which is essential for transferring image pixel data. The CSI interface is connected to the pi using a 15-pin ribbon cable.

b) UART

The UBlox Neo-6M GPS Module will be interfaced serially with the raspberry pi using manual wiring. The connection can be seen in Figure 2. The GPS module also comes with an external U.FL ceramic antenna for better reception. The manual Bluetooth control of the system also employs UART for the serial communication between the robot motors and the Bluetooth control android application.

c) USB

The raspberry pi and Arduino require a connection to a power source via a USB connection.

3.1.4 Software interfaces

The entire system was developed using the following software libraries;

- **Raspbian**

Raspbian is the native operating system for the raspberry pi, which is the system's primary controller. Raspian is built on top of the Linux OS kernel.

- **TensorFlow**

A supervised learning classifier will be created using the TensorFlow library. The TensorFlow library is conducive for deep learning applications and machine learning. TensorFlow was used in training the convolutional neural network model for classifying images into diseased and healthy. The different architectures to be explored are the MoblieNet, ResNet, and LeNET CNN architectures. Each of these architectures has its strengths and weaknesses, and the most efficient of them would be deployed in the final system.

- **TensorFlow Lite.**

TensorFlow Lite is a library of tools that can run TensorFlow models on embedded boards and IoT devices. The TensorFlow lite library enables low-power devices to run machine learning inferences with low latency and quantized binary size. This software interface is very critical to the premise of this project; Machine learning on embedded boards.

- **OpenCV**

OpenCV is an open-source Python library used for image processing, video processing, and computer vision. The disease detection model created using tensor flow lite will receive its input tensors through the OpenCV library. The OpenCV library was also employed in performing specific data processing tasks like image segmentation and data augmentation for training the

TensorFlow model. This library will also be used in the development of a disease quantifier script. This script will use pixel data from the camera to determine what percentage of the leaf is damaged.

- **Firebase Database**

The Firebase database is an adaptable database that is conducive for growing data like the image feed this system produces.

- **MIT app inventor**

MIT App Inventor is a web application IDE created by Google and is maintained by the Massachusetts Institute of Technology. App inventor was used as a development environment for making the android application that facilitates the manual Bluetooth control of the robot.

3.1.5 Communications interfaces

The following communication protocols were used in the system:

HTTP: The hypertext transfer protocol will be employed to transmit image metadata such as time and location to the database.

FTP: The Raspberry pi uses the file transfer protocol to transmit images of diseased leaves to the database.

TCP/UDP: The Raspberry Pi is equipped with an onboard 802.11n Wireless LAN adapter which uses to transmit a video feed from the Pi camera Rev3 to the controlling system

Bluetooth: The HC-05 Bluetooth module was interfaced with the Arduino for serial communication between the Arduino and the android control application.

3.1.6 Operations

The disease detection system will consist of two subsystems.

- Image acquisition and positioning system
- Robotic system.

Image Acquisition and Positioning Subsystem.

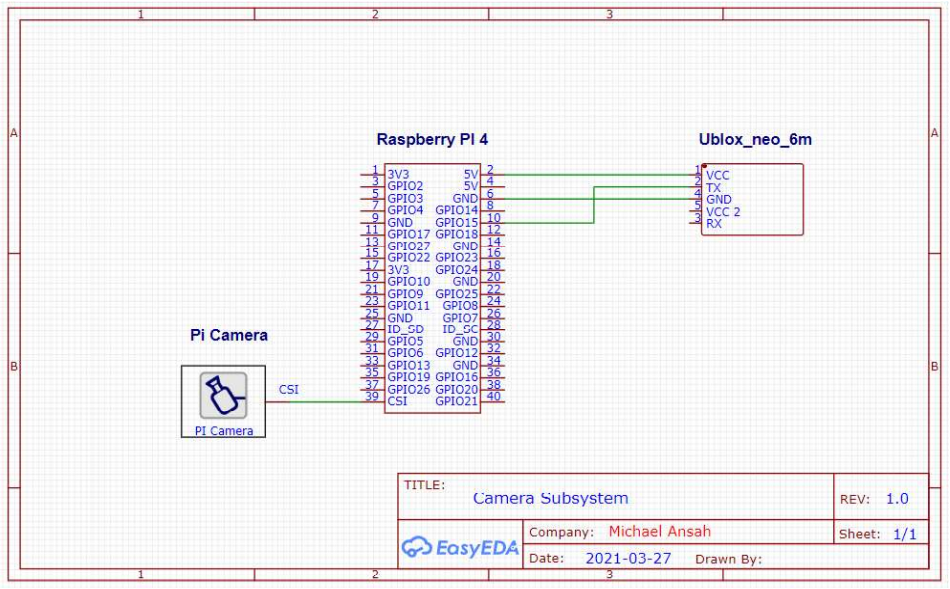

Figure 2: Schematic Diagram Of Image Acquisition and Positioning Subsystem.

The image acquisition and positioning subsystem, as shown in Figure 2, will be responsible for capturing images into the system, running the disease detection algorithm on the image, and transmitting the image and the results of the algorithm to the database. This system will also be responsible for sending the location of the diseased leaf to the database.

Robotic subsystem

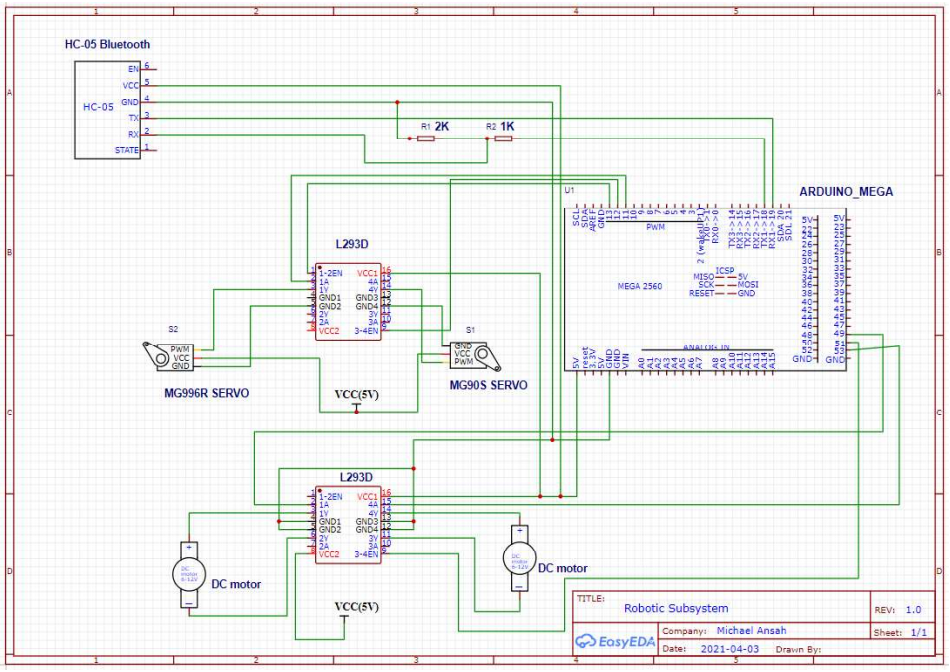

Figure 3: Schematic Diagram of Robotic Subsystem

The Robotic subsystem will be responsible for traversing the farm with the camera and pointing it at its area of interest. As shown in Figure 3, the robot vehicle chassis is powered by two DC motors. Two additional servo motors power the camera rotation actuator. These motors are interfaced with an Arduino Mega by using the Adafruit Motor shield/L293D. The HC-05 Bluetooth module was used to control the motor by communicating with the Arduino Serially.

There are potential solutions that could provide the user with some data to help manage the blight disease. The different methods have distinctive drawbacks and advantages.

The two main methods considered for blight disease detection were image processing and machine learning. The machine learning method would involve training a neural network model using a dataset of images of the blight disease symptoms. This model will make a diagnosis using what it has learned from the dataset training.

The second option would be to use machine learning reinforced with image processing. After the machine learning model makes its prediction, the pixel data from the image will deduce more insights into the disease symptoms. In this case, the pixel data will be used to ascertain the extent to which the leaf is damaged.

In Table 1 below, these two approaches assed how well they meet the user's needs. This analysis showed that the Machine learning model reinforced with image processing is the best way to go. The pugh matrix proves that this approach addresses the user's need better than solely using machine learning. The Criteria of the pugh matrix were given weights depending on their relevance to the user. For a disease detection system, accuracy is the most critical metric. For this reason, it was assigned the highest weight of 4.

Criteria	**Baseline**	**Weight**	**A(Solely Machine learning)**	**B(Image Processing & Machine Learning Model)**
Safety	0	3	1	1
Ease of Set Up	0	1	0	0
Reliability	0	4	0	1
Speed	0	3	1	1
Usability at night	0	1	-1	0
Accuracy	0	4	1	1
Start Up Cost	0	2	-1	-1
Maintenance cost	0	2	-1	-1
TOTAL		20	5	10

Table 1: Pugh Chart For System Feature Selection. The baseline represents disease identification by human judgment. Concept A represents a system that detects the leaf blight disease using

only a pre-trained convolutional neural network. Concept B represents a system that detects the leaf blight disease using a pre-trained convolutional neural network model and an image processing script that quantifies the damage done to the leaf.

In light of this choice of approach, the system would operate in the following steps. The camera live feed and robot are initialized. The machine learning model will then perform inferences of every frame of the camera live feed. The moment model detects a diseased leaf in the camera frame; an image is captured. The image is run through an image processing script to quantify the damage done to the leaf. The results of the image processing script, the image, the GPS coordinates of the robot at the time of the diagnosis are sent to the database. The data is then conveyed to the user through the web app. Figure 4 presents these steps pictorially.

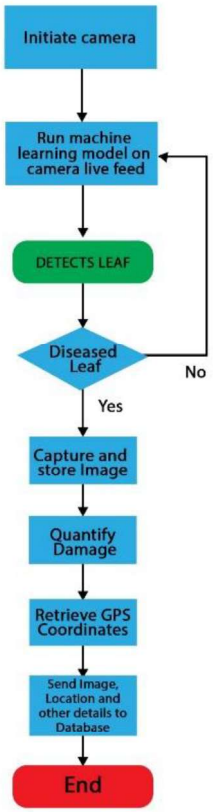

Figure 4: A flow of the operation of the disease detection system.

3.1.7 User Requirements

The primary users for this project are farmers, greenhouse nursery workers, and technicians. In totality, the system should meet the needs of these primary users. The requirements are stipulated below:

1. The user should be able to view all diseased leaf incidents.
2. The user should be able to view what the robot sees through a live feed
3. The user should receive an alert when a diseased leaf is detected
4. The user should be able to discern the extent to which the leaf is damaged

3.1.8 System Requirements

To meet the user's requirements and truly fulfill its function, the system must meet the following requirements.

1. The system should capture images of leaves from all angles from the robot's current position.
2. It should autonomously classify a leaf as diseased or healthy.
3. It should traverse paths on a farm.
4. It should identify the location of diseased leaves.
5. It should quantify the damage of the leaf from the image pixel data.
6. The system should not be bulky.
7. The system must be able to report what time the image was captured.
8. All captured images and disease information should be accessible using the dashboard.
9. The system should be able to notify the user of any mechanical fault.
10. The system should have internet connectivity to communicate with the database.

3.2 Neural Network

The machine learning models explored for the study were convolutional neural networks because of their inherent ability to create feature maps. The model uses feature maps to learn trends in abstract shapes and edges in image data. These features can be pretty complex. Hence the multilayered architecture of neural networks is conducive for this project. The convolutional neural networks were built using the TensorFlow and TensorFlow Lite libraries. The TensorFlow library was used to create the initial convolutional neural network models. After which, these models will be quantized so that they can run on embedded boards. The quantized versions will be created using the TensorFlow Lite library flashed onto our target platform. The library will also create an environment in Python that will run these models on a raspberry pi.

The convolutional neural network models in question for this project are the LeNet, MobileNet, and ResNet architectures. These different architectures were explored for their inherently different features and use cases. However, only the architecture with the best results, MobileNet V2, will be discussed.

MobileNet CNN Architecture.

The Google research team developed the MobileNet architecture. They have a streamlined architecture that uses depth-wise separable convolutions to build lightweight deep neural networks (Howard, et al., 2017). These lightweight neural networks are made purposely for deployment on mobile devices and embedded devices.

3.3 Damage Classifier Design.

After the neural network identifies a diseased leaf, the system captures an image of the leaf and runs it through an image processing script written in Python with the OpenCV library. This script assigns a numerical value to the amount of damage done to the leaf using pixel data. The tomato blight disease is characterized by brown patches on the leaf surface. The script uses OpenCV functions to isolate these brown patches and calculate their total area as a percentage of the total area of the leaf. Figure 5 shows the steps involved in this process. This percentage quantifies the damage done to the leaf. Equation 1 expresses the mathematical formula of this percentage.

$$Damage(\%) = = \frac{\boldsymbol{Area\ of\ Diseased\ Contour}}{(\boldsymbol{Area\ of\ Healthy\ Leaf\ Contour} + \boldsymbol{Area\ of\ diseased\ contour})}$$

Equation 1: Leaf Damage Quantification Formula

The main methods employed in the image processing script were color masking and image thresholding. A threshold is a form of image segmentation that converts grayscale image pixels to black and white based on its brightness value of each pixel. (Quweider, Scargle, & Jackson, 2007). Color masking involves isolating pixels that fall within a specific color range. Thresholding and color masking was used concurrently to isolate the areas of interest of the leaf; The healthy green surface and the brown patches of the leaf.

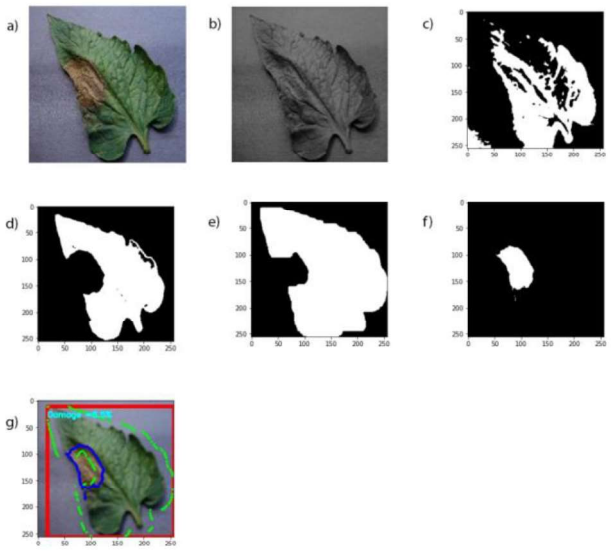

Figure 5: Breakdown of the steps taken to quantify leaf damage. a) The original image of a leaf with blight disease. b) The image is converted to a greyscale image. c) The image is thresholded. d)The green pixels of the original image are converted to white, and every other pixel is converted to black. e) the results of step c) and e) are combined by a bitwise and operation. f) Brown pixels are converted to white, and every other pixel is converted to black to isolate the diseased patch. g) The area of the brown patch is expressed as a percentage of the total area of the leaf.

3.4 Application Software

The web application has a dashboard. This dashboard conveys the data produced by the system to the user, as shown in Figure 6. The data is presented in the form of a line chart. The software also generates a report in a pdf format which gives a succinct overview of the disease situation on the farm. Farmers can send this report to their extension officers. If many farms adopt this system, there will be enough data to inform decisions on a regional level. As shown in Figure 7, the dashboard alerts the user when the system detects a new disease and a report is ready to be downloaded.

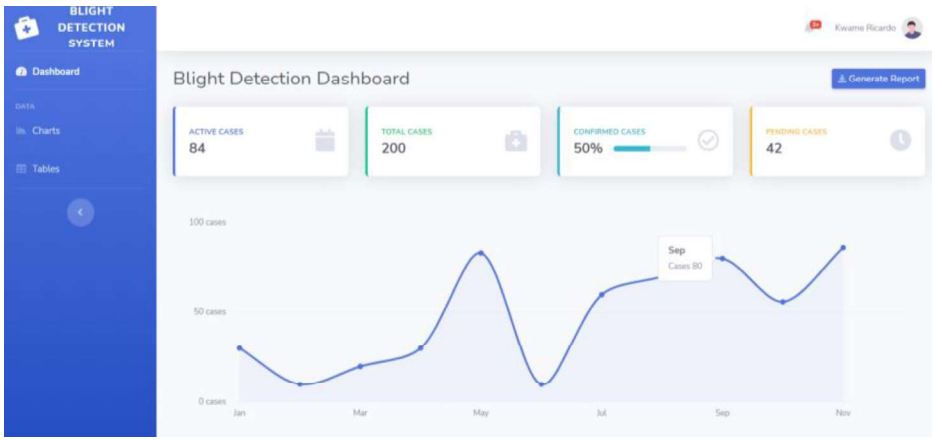

Figure 6: Screenshot of the Dashboard interface. The dashboard shows the number of active cases detected by the system, the pending cases waiting to be confirmed, and the total cases detected by the system.

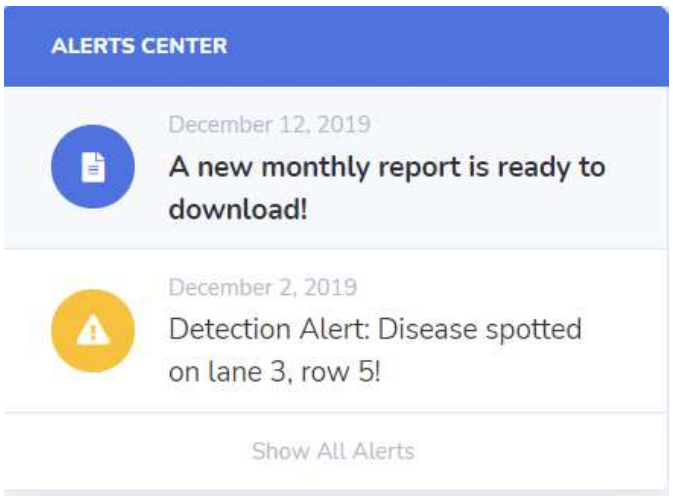

Figure 7: Notification panel showing the disease detection system alerts

3.5 Database

The database's primary purpose is to store details concerning the disease detections on the farm for conveyance by the web app. The selected database for the system is Firebase. Firebase is a backend application development software that can be used to create non-relational databases. The data is not cumbersome and does not have a lot of relational features. Hence the flat hierarchical nature of Firebase schemas is ideal for this project. Relational databases like SQL use tables to store data. These tables are inflexible compared to NoSQL databases like Firebase. Firebase Schemas give the database designer the liberty to define how the data is organized.

The data structure for Firebase is a JSON file. JSON files have keys and values. The two entities in this database are cases and alerts, as shown in figure 8. Each case has four keys; The case number, the case status, leaf damage location. The case number is the unique identification number given to every new disease detection. The case status is a Boolean that communicates whether the disease detection has been a confirmed case of tomato blight and not a false positive.

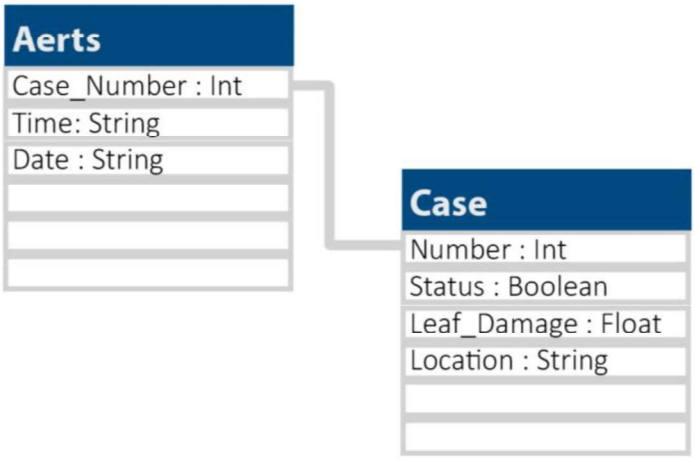

Figure 8: Database Design

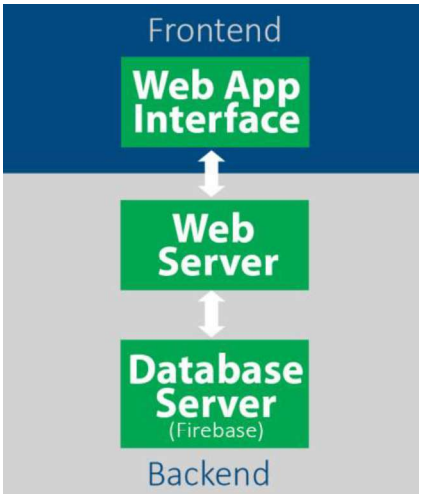

Figure 9: High-level application software architecture

Figure 9 illustrates the architecture of the software system. The arrows indicate the directions in which each component of the system interacts with the other. The user interacts with a web application that will have hosting on a web server. The web server will communicate with the database server to retrieve and modify data concerning the disease detections.

3.6 Constraints

The system has some limitations. They are highlighted below:

- **Light intensity**

  The image machine learning model performs poorly in low-light. Hence the light intensity largely affects the effectiveness of the disease classifier. A possible solution would be to purchase a camera that has a sensor with higher light sensitivity. Such a sensor would be able to capture the features of interest despite the low light condition. An entirely new model might also have to be trained using images captured in low light

- **Shadows**

  The camera's position relative to the source of light and the leaf may result in shadows being cast on the leaf. The model may confuse the shadows as diseased spots. This confusion may ultimately lead to inaccurate predictions.

- **Motion blur & Rolling shutter**

  When cameras are in motion, the images captured would have motion blur. The severity of the blur depends on the velocity with which the camera is in motion. Motion

blur distorts the image, and the model may not be accurate on a distorted image. Rolling shutter is also a phenomenon that distorts the camera image when fast-moving objects are in the camera frame or when the camera moves at high speed. The camera images are vulnerable to rolling shutter distortion when the robot is moving at a considerable speed.

- **Clustered Leaves**

  The machine learning model classifies one leaf at a time because the model was trained with images of individual leaves. This means the model will not work very well with clustered leaves in the frame. The damage quantifier is also likely to confuse the overlapping contours of multiple leaves as one leaf.

3.7 Alternative Considerations

Alternatively, this system could be implemented using the following approaches;

- **Computer vision boards**

A computer vision board could have been used for the system instead of a microcomputer. Prime examples of computer vision boards are the Openmv H7 and the OpenCV artificial intelligence kit. They are a few of the many embedded boards made specifically for computer vision. Most of these boards have a camera built-in with replaceable lenses. Replaceable lenses make the system more versatile because different lenses can be installed on the camera for other use cases. The downsides of using such boards are their

limited processing power and limited ports for connecting more accessories. They are often not as computationally powerful as microcomputers like the raspberry pi.

- **Surveillance Setup**

Another alternative means of implementing the system would be to install numerous cameras at specific surveillance points on the farm instead of having a robot traverse the farm with a camera. This setup may provide a more wholesome surveillance method. Still, it would be very costly because purchasing enough cameras to provide total coverage of the farm will be far more expensive.

- **Smartphone.**

The disease detection algorithm could be deployed on a smartphone. This provides some measure of convenience because smartphones are commonplace in this age of technological advancement.

Chapter 4: Implementation and Methodology

4.1 Hardware Setup

This system consists of a raspberry pi camera Rev 1.3, Ublox-Neo 6m GPS module, and the Raspberry pi 4 single-board computer. Both the Pi Camera Rev 1.3 and the Neo 6M GPS module will be interfaced with a raspberry pi 4. The raspberry pi 4 will play the primary role of processing image data and GPS data and communicating wirelessly with the database. It is adequately equipped with a Quad-Core Cortex-A72 (ARM v8) processor and 4GB of RAM for this task. The raspberry 4 will run the TensorFlow lite model, which will receive data input from the pi camera. The model will make inferences on the image data received from the camera and save images with diseased leaf detected into the online database via Wi-Fi. The GPS module does real-time location tracking, and the raspberry pi will transfer images of the diseased leaf coupled with metadata concerning the location of the leaf of the farm and the time the image was captured.

4.2 Software

4.2.1 Front-end Web interface

The web application serves as means for Farmers, extension officers, or any other administrative stakeholders to view the diseased leaves, their location on the farm, as well as their diagnostic results. The user can mark each detection entry as attended to or not. The web interface was created using HTML and CSS. Figures 6 and 5 show screenshots from the front-end web interface.

4.2.2 Backend system

The tomato blight disease detection algorithm is written in Python using the TensorFlow, TensorFlowlite, and OpenCV libraries. A major component of this program is a custom-trained image detection model using the Mobilenet SSD architecture. The model reads its input from the live feed of the camera. This live feed is temporary because it is stored on the raspberry pi RAM. RAM is volatile, temporary memory. When a leaf with blight symptoms is detected, a picture is taken; this picture is stored on the microSD card of the raspberry pi. This storage is non-volatile, and the photo will remain on the device until it is deleted. Using the pixel data of this stored image, the extent of the disease damage to the leaf is quantified. The location of the leaf, the damage details, the time the picture was taken is sent to the database, and the farmer is notified.

4.3 Rolling shutter Phenomenon

In cameras with CMOS sensors, images are captured line by line on the sensor instead of being captured all at once. The sensor scans the image in rows from top to bottom. The top of the image frame is recorded slightly earlier than the lower frames, as illustrated in figure 10. Camera sensors that capture information this way are more efficient and less prone to overheating (Liu, Guan, & Wen, 2019). However, this delay may result in image distortions when fast-moving objects are being captured or when the camera itself is in motion. This phenomenon is known as the rolling shutter effect. Figure 11 shows the distortion caused by this effect.

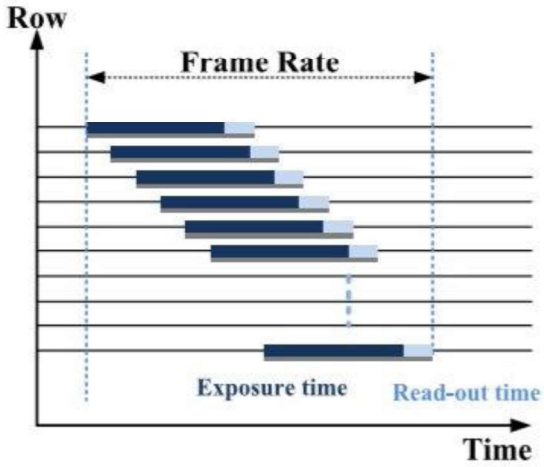

Figure 10: Schematic Of Rolling Shutter Effect (Liu, Guan, & Wen, 2019)

Two types of modern Image sensors exist CMOS and CCD sensors. CMOS sensors are relatively affordable, and they are used in everyday cameras like smartphone cameras and digital cameras. They are prone to the rolling shutter effect. Cameras with a CCD sensor capture an image in its entirety all at once. However, CCD sensor cameras are vastly more expensive and difficult to manufacture, unlike CMOS sensor cameras which capture images without emitting excess heat and draining a lot of battery power. (Liu, Guan, & Wen, 2019)

One of the main hallmarks of this leaf disease detection system is cost-efficiency. This is why a camera with a CMOS sensor(Pi camera Rev 3) was selected for the project.

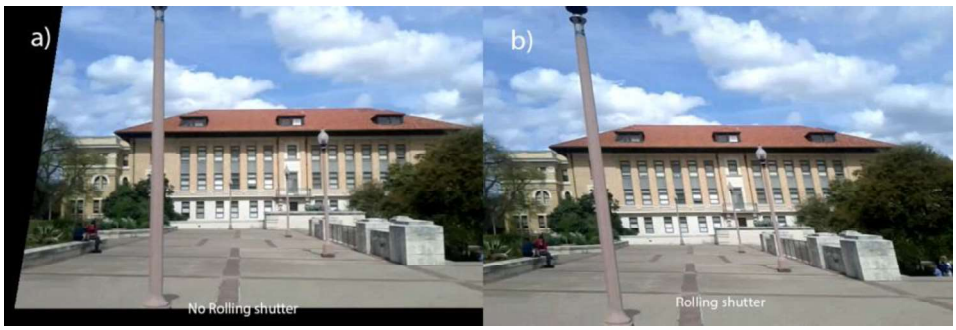

Figure 11: a) Digitally corrected Image with no rolling shutter effect . b) Image with rolling shutter distortion(on the right). (Jia & Evans, 2012)

4.4 Data Preparation and Model Training

PlantVillage provided the image dataset on Kaggle. The dataset consists of images of Healthy tomato leaves and tomato leaves with different categories of diseases. The leaf disease in question for the study is early and late blight. For this disease, there were 1591 images of healthy leaves and 1909 images of diseased leaves. After manually culling the images for irregularities that would affect the model, the final data set had 1453 Images with a distribution of 694 Images of Healthy leaves and 759 images of diseased leaves.

The premise of this project is to explore the deployment of compressed machine learning models on small embedded devices. Embedded devices have limited computational power and storage. Hence, having a highly optimized model is imperative. Different dataset sizes were used to train the model and assess its performance to explore the trade-off between accuracy and model size.

Three different-sized datasets were used for this experiment. Each of them maintained the same ratio of training images to test images; 2:5.

Dataset	Training Images	Test Images
Large	1046	407
Medium	513	199
Small	117	46

Table 2: Dataset Image distribution. This table shows the number of Images in each dataset used in the experiment.

To select the best model architecture for the experiment, different dataset variations were trained using 3 model architectures; The MobileNet V2 Architecture, the LeNet architecture, and the ResNet50 architecture. Each of the models was trained on the tomato leaf blight dataset to assess their performance. A model is said to converge when it begins to make accurate predictions (Warden & Situnayake, 2019). Among the 3 of the models, the MobileNet V2 architecture showed the best convergence.

The efficacy of models can be assessed by analyzing two parameters; Loss and Accuracy. Loss provides an estimate of how much further a model is away from being perfect. Accuracy depicts the correct predictions the model makes as a percentage. Pete Warden & Daniel Situnayake purport that a perfect model would have a loss of 0.0 and an accuracy of 100%. (Warden & Situnayake, 2019).

The model input size was specified at a width of 320, a pixel height of 320, and 3 color channels. This size was selected because the 320 by 320 image resolution is roughly half the of the 640 by 480 resolution camera feed captured by the pi Rev 1.3 camera. With a smaller input size than the native camera resolution, the system performs fast, and latency is minimized. A

significant feature of the tomato blight disease symptom is the discoloration of the leaf. Hence colored images (images with three color channels) were used to train the model.

The basic unit of a convolutional neural network is a neuron. Neurons compute weighted sums of their input to produce an output, as shown in Equation 2. The output value of the neuron is the basis on which the classification is made. The function that determines the output of a neuron is called an activation function.

$$Output = \sum(Weight * Input) + Bias$$

Equation 2 : General neuron activation function

Softmax was selected as the last activation function of the neural network. The last activation function of the neural network determines the final output of the entire model. Softmax was selected because it returns a multinomial probability as a final result. In simple terms, softmax returns the probability of the input data belonging to a specific class. In terms of this project, softmax returns the probability of a leaf having blight symptoms. The softmax activation function does this by taking an input vector and normalizing the distribution of vectors into probabilities that sum up to one.

$$\sigma(z^{\rightarrow})_i = \frac{e^{zi}}{\sum_{j=1}^{K} * \ e^{zi}}$$

Equation 3: Mathematical Expression for softmax Function

$z^{\rightarrow}$ = Input vector of the softmax function

zi = Elements of the input vector to the softmax function

$\sum_{j=1}^{K}$= Normalization term

e^{zi}= Input vector exponential function.

K = number of classes.

The mobile V2 architecture performed well in learning the data and predicting images in the test set. It converged well after ten epochs. The model validation accuracy was higher than its training accuracy at every epoch. This proves that the model could generalize its learning onto data that it has not seen before. The validation loss was also lower than the training loss at every epoch. Figure 12 shows the convergence of the model.

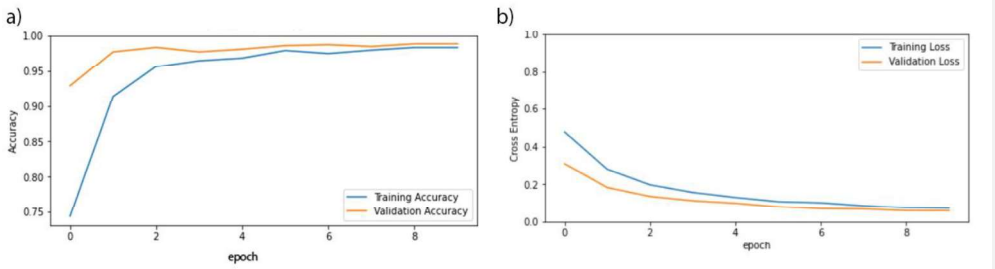

Figure 12: MobileNet V2 architecture training results. a) Accuracy b)Loss. The model was trained with two classes; Healthy Leaf and Diseased Leaf. The training was carried out for 15 epochs at a learning rate of 0.001. Input size=(320,320,3). Activation function=softmax. The model was trained three times to ensure consistent performance.

Chapter 5: Results

5.1 Results.

In the previous chapter, we discussed an experiment to answer the question; Does the number of images in the dataset affect the model accuracy? After training models using the respective datasets, their accuracies and losses with each epoch were averaged, as seen in Table 3. The large dataset yielded the best results as the model had the highest average accuracy (82%) and the lowest average percentage loss(39%).

Dataset	**Large**	**Medium**	**Small**
Mean Accuracy(%)	82	78	63
Mean Loss(%)	39	58	50
Mean Error	0.179	0.1722	0.0414

Table 3: Accuracy and loss results for Large, Medium, and Small Datasets.

To validate the results of the experiment, an ANOVA with posthoc Tukey pairwise comparison was performed to assess whether the accuracies of the datasets were statistically significant from each other. The results are illustrated in Figure 13 and Table 4. The result of the Tukey pairwise comparison proved that the models trained with a larger dataset yielded a higher accuracy. Figure 13 denotes the statistical difference between the accuracies of all the datasets. This proves that the trade-off between dataset size and model accuracy still exists on edge devices that run compressed models. In this light, the supervised learning model for this project had to be trained with enough images that would yield high accuracy.

Comparison Pair	P-Value
Medium-Large	0.0016782
Small-Large	0.0000002
Small-Medium	0.0162566

Table 4 : P-Values from ANOVA. The p-values for the Medium-Large and the Small-Medium pairs were less than 0.05; this implies each pair had a statistical difference. The small-Large pair had a highly statistical difference with a p-value less than 0.001

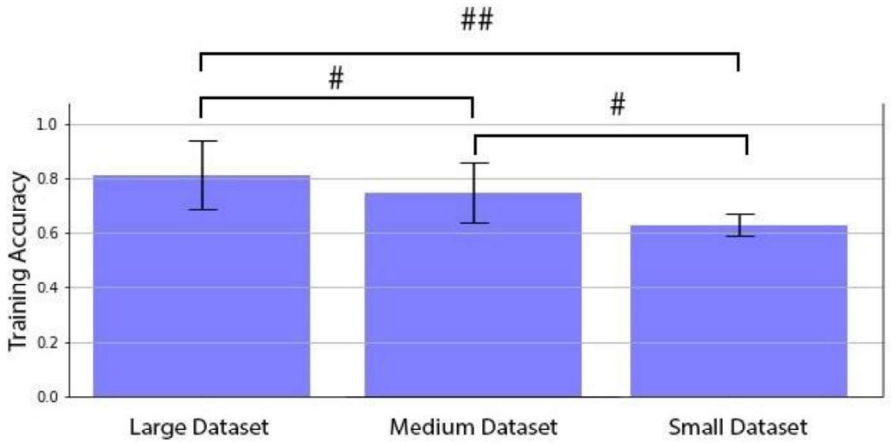

Figure 13: MobileNet V2 architecture training results. a) Accuracy b)Loss. The model was trained with two classes; Healthy Leaf and Diseased Leaf. The training was carried out for 15 epochs at a learning rate of 0.001. Input size=(320,320,3). Activation function=softmax. The model was trained three times to ensure consistent performance. All datasets had a statistically significant difference between each other—pairs with # show statistical significance between them. ## shows a high statistical significance.

5.2 Motor speed.

The system has Bi-directional DC motors that propel the robot vehicle forward. The motor speed is directly proportional to the voltage supplied. A total of 9 volts was delivered to each motor. The robot vehicle must be able to traverse the farm quickly but not too fast for the

camera to capture images properly. On average, the robot vehicle travels 1m in 2.64 ± 0.0518 seconds.

5.3 GPS data transfer speed.

It is imperative that the GPS module updates the location of the robot promptly. This would prevent any wrong coordinates from being transmitted for diseased leaf sitings. The Ublox Neo-6M GPS module transfers data in pulses then and updates them at a regular frequency. The time in between each coordinate update was recorded and tabulated below. The average speed of the GPS update is 1.018 ± 0.13 seconds

5.4 Camera image fidelity

To assess the robustness of the camera image quality in different lighting conditions. The camera was tested in 3 different lighting conditions; Daytime, nighttime, and an artificial light source at night. Two leaves were used as image subjects. One leaf was painted with dark oil paint to mimic a blight spot, and the second leaf was a healthy leaf.

Figure 14: Screenshots from camera fidelity experiment. Two sets of images were captured and run through the supervised learning model under three different lighting conditions; a)Daytime, b) Nighttime c) Nighttime with an artificial light source.

Ten sets of images were captured for each of the two classes of leaves under three different lighting conditions, as shown in Figure 14. The inference time and the multinomial probability result from the softmax function (as discussed in chapter 4) are displayed. The inference times, probability, and predictions were recorded as shown in Tables 5,8, and 11. To reiterate, this procedure was carried out to test how the system performs under different lighting conditions.

Subject: Healthy Leaf (Correct result=Negative**)**				**Subject : Diseased Leaf (Correct Result =** Positive**)**			
Image No	**Probability**	**Inference Time (milliseconds)**	**Result**	**Image No**	**Probability**	**Inference Time (milliseconds)**	**Result**
1	1.00	66.6	Negative	**1**	0.96	50.3	Positive
2	0.96	49.8	Negative	**2**	0.97	51.8	Positive
3	0.95	50.2	Negative	**3**	0.97	50.6	Positive
4	0.90	50.6	Negative	**4**	0.98	53.2	Positive
5	0.99	51.0	Negative	**5**	0.97	50.5	Positive
6	0.89	50.3	Negative	**6**	0.93	53.5	Positive
7	0.88	50.8	Negative	**7**	0.94	49.9	Positive
8	0.85	50.9	Negative	**8**	1.00	50.2	Positive
9	0.92	50.2	Negative	**9**	0.93	50.5	Positive
10	0.99	50.2	Negative	**10**	1.00	50.7	Positive
AVG	0.933 ± 0.05	52.063 ± 5.12ms		**AVG**	0.965 ± 0.026	50.72 ± 1.175ms	

Table 5: Daytime prediction results. The supervised classifier was tested on ten captured images each of diseased leaves and healthy leaves in daylight. In the daytime, the classifier was 100% accurate and all the predictions made were correct. The classifier also performs relatively fast in the day, with an average inference time of 52.063 ± 5.12ms. The multinomial probability of each classification is also recorded in the table.

No. Images= 20	**True Positive**	**True Negative**	**Total**
Predicted Positive	10	0	10
Predicted Negative	0	10	10
Total	10	10	

Table 6 : Confusion Matrix for Daytime Images. Positive = Diseased Leaf. Negative = Healthy Leaf

Measure	Value
Sensitivity	1.0
Specificity	1.0
Precision	1.0
Negative Predictive Value	1.0
False Positive Rate	0.0
False Discovery Rate	0.0
False Negative Rate	1.0
Accuracy	1.0

Table 7: Daytime Confusion Matrix Results. Sensitivity refers to the rate at which the predictor makes true positive predictions. Specificity is the rate at which the true negative predictions are made. Precision is a measure of the positive predictive value of the classifier.

Subject: Healthy Leaf (Correct result=Negative)				**Subject: Diseased Leaf (Correct result=Positve)**			
Image	**Probability**	**Inference Time (milliseconds)**	**Result**	**Image**	**Probability**	**Inference Time (milliseconds)**	**Result**
1	0.70	48.2	Negative	**1**	1.00	50.9	Positive
2	1.00	50.3	Negative	**2**	0.71	50.7	Negative
3	0.96	50.4	Negative	**3**	0.98	56.3	Positive
4	0.89	52.3	Positive	**4**	0.57	50.1	Negative
5	0.86	50.5	Negative	**5**	0.90	50.9	Positive
6	0.76	50.5	Positive	**6**	1.00	53.5	Negative
7	0.95	50.5	Negative	**7**	0.94	49.9	Positive
8	0.72	50.4	Positive	**8**	1.00	50.2	Positive
9	0.86	53.2	Positive	**9**	0.55	50.5	Negative
10	0.89	50.5	Positive	**10**	1.00	50.7	Positive
AVG	0.859 ± 0.103	50.68 ± 1.314ms		**AVG**	0.865 ± 0.183	51.37 ± 2.002ms	

Table 8: Nighttime prediction results. The supervised classifier was tested on ten captured images each of diseased leaves and healthy leaves at night time. The classifier was 55% accurate, and 9/20 of the predictions made were false. The classifier also performs relatively fast at night with an average inference time of 50.068 ± 1.314ms.

No. Images= 20	**True Positive**	**True Negative**	**Total**
Predicted Positive	6	5	10
Predicted Negative	4	5	10
Total	10	10	

Table 9: Confusion Matrix for Nighttime Images. Positive = Diseased Leaf. Negative = Healthy Leaf

Measure	**Value**

Sensitivity	0.6
Specificity	0.5
Precision	0.55
Negative Predictive Value	0.55
False Positive Rate	0.5
False Discovery Rate	0.45
False Negative Rate	0.4
Accuracy	0.55

Table 10: Nighttime Confusion Matrix Results. Sensitivity refers to the rate at which the predictor makes true positive predictions. Specificity is the rate at which the true negative predictions are made. Precision is a measure of the positive predictive value of the classifier.

Subject : Healthy Leaf (Negative)				**Subject : Diseased Leaf (Positive)**			
Image	**Probability**	**Inference Time (milliseconds)**	**Result**	**Image**	**Probability**	**Inference Time (milliseconds)**	**Result**
1	0.83	123.5	Positive	**1**	0.96	99.2	Positive
2	0.75	124.0	Positive	**2**	1.00	102.0	Positive
3	0.93	124.2	Negative	**3**	0.99	99.5	Positive
4	0.98	124.7	Negative	**4**	0.90	100.4	Positive
5	0.82	123.0	Negative	**5**	0.93	104.8	Positive
6	1.00	123.3	Negative	**6**	1.00	105.1	Positive
7	0.99	123.1	Negative	**7**	0.99	100.9	Positive
8	1.00	123.5	Negative	**8**	0.81	89.8	Positive
9	0.98	51.5	Negative	**9**	0.90	50.4	Positive
10	0.96	76.8	Negative	**10**	0.88	81.3	Positive
AVG	0.924 ± 0.09	111.76 ± 25.80ms		**AVG**	0.936 ± 0.063	93.34 ± 16.75ms	

Table 11: Nighttime Images with external light prediction results. The supervised learning classifier was tested on ten captured images of diseased leaves and healthy leaves at night with an artificial light source introduced. The classifier was 90% accurate, and 2/20 of the predictions made were false. The classifier also performed relatively slower with average inference times of 111.76 ± 25.80ms and 93.34 ± 16.75ms.

No. Images= 20	**True Positive**	**True Negative**	**Total**
Predicted Positive	10	2	10
Predicted Negative	0	8	10
Total	10	10	

Table 12: Nighttime Images with external light Confusion Matrix. Positive = Diseased Leaf. Negative = Healthy Leaf

Measure	Value
Sensitivity	1.00
Specificity	0.80
Precision	0.83
Negative Predictive Value	1.00
False Positive Rate	0.20
False Discovery Rate	0.17
False Negative Rate	0.00
Accuracy	0.90

Table 13: Nighttime with external lights Confusion Matrix Results. Sensitivity refers to the rate at which the predictor makes true positive predictions. Specificity is the rate at which the true negative predictions are made. Precision is a measure of the positive predictive value of the classifier.

5.5 Discussion

The confusion matrices results show that the system performs best in daylight with an accuracy of 100% and a false discovery rate of 0%. The system performs poorly in night light with an accuracy of 55% and a false discovery rate of 45%. However, this poor performance during nighttime can be compensated for using an external light source to light the subject. The accuracy at nighttime when an external light was used was increased to 90%. Despite the increased accuracy, the system performed roughly two times slower than in the other lighting conditions, with an inference time of approximately 111.76 ± 25.800ms.

In any disease classifier, the most detrimental predictions are the false-negative predictions. This is because a potentially malicious disease has been written off as harmless. False-negative predictions will lead to devastating effects because the farmer thinks no diseases have affected the farm crops when that is not the case. The infection will spread and cause more damage when the farmer is under the false impression that his farm is disease-free. As such, false-negative predictions must be minimized entirely in the system. The false-negative

rate of the system was at its minimum at 0% when the classifier was used during the day. The false-positive rate was at its highest at 40% when the classifier was used in nightlight.

In conclusion, the fidelity of the camera images is gravely compromised in night light. The system should be used only during the day to avoid the devastating effects of false predictions. Under the urgent circumstance where the user needs to use this at night, an artificial light source must be used with the camera. However, the classifier performs two times slower.

Chapter 6: Conclusion

6.1 Summary

In summary, this project explores the detection of leaf diseases using image pixel data and machine learning. A robot car vehicle chassis is used to convey the camera around the farm, and a camera is used to capture images that would be fed into the model. Positive cases detected by the model are run through a Python script quantifying the damage done to the leaf using pixel data and expressing it as a percentage. The image of the diseased leaf, the date and time the image was taken, the location of the robot at the time of the image capture, and the percentage damage of the leaf are then transmitted to a database. The user can access this information using a dashboard application. In a nutshell, the system surveys a farm and detects tomato leaves with lesions and marks that are symptomatic of the tomato blight disease.

The machine learning model is deployed on a raspberry pi which is interfaced with the pi camera rev 1.3 and the Ublox neo 6M GPS module. The Robot vehicle is propelled by two bi-directional DC motors(one for each wheel). The camera can tilt up and down and pan left and right. Two servo motors control the pan and tilt functionalities of the camera. All the motors in the system are controlled using an Arduino mega interfaced with an ardafruit motor shield.

The study answers the following questions. Can machine learning models perform satisfactorily on edge devices? Does the dataset size affect the performance of the model? Is the performance of the model affected by any external stimuli?

6.2 Results

The study does indeed prove that machine learning models can perform satisfactorily on edge devices. The final model used for the system had a size of 16.1MB, and it was able to make inferences in time as quickly as 50 milliseconds.

The dataset size affected the model's performance, and the trade-off between model size and accuracy proved unavoidable. The original model was training using 1046 images, and it achieved an average validation accuracy of 84% over 15 epochs of training. The validation accuracy of the model was reduced by 19% when only 10% of the dataset was used to train(117 images) the model. The validation accuracy of the model dropped by 4% when half of the dataset(513 images) was used to train the model.

The robustness of the model under different lighting conditions was put to the test in an experiment. A total of 60 images were captured under different lighting conditions to assess the model's performance during different times of the day. In the daytime, the model predicted expected results with an accuracy of 100% and 0 false predictions out of 20 images. At night time, the model performed poorly with an accuracy of 55% and 11 false predictions out of 20. An artificial light source was introduced at night, and the accuracy increased to 90%, with the number of false predictions dropped from 11 to only 2.

6.3 Limitations.

Overall the study proved to be a sound proof of concept of the potential possessed by edge computing. However, the system had a few pitfalls; the model could only make classifications with one leaf in the frame. This would prove to be a major bottleneck on an actual farm because leaves naturally occur in clusters. The pi camera rev 1.3 has a small CMOS sensor which does not perform too well in low light. As mentioned earlier, this gravely limited the performance of the system in low light. This lapse in image clarity could have been improved by using a camera with a more sensitive sensor. Such a sensor would produce images with more discernable features even in low light.

Another limitation was with the leaf disease quantifier. The quantifier uses the colors in the image to make a prediction. With varying lighting conditions, the color of the leaf may appear different from what the algorithm is intended to read. This occurrence was very common during nighttime because shadows were present and different shades of green were also present. Figure 15 shows how the algorithm isolates only a portion of the leaf because of the color difference caused by shadows. The algorithm would have to be tweaked to detect green color values within this new range to solve this. A possible solution to this problem would be to train a separate machine learning model that predicts green color values that would create a suitable mask for the quantification algorithm.

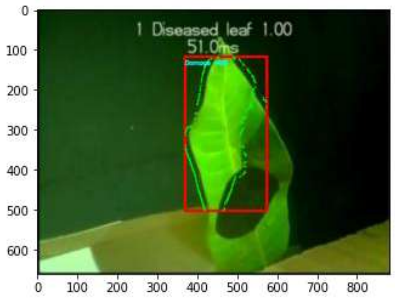

Figure 15: Disease Quantifier masking only part of the leaf. The algorithm mistakes only a portion of the leaf as an entire leaf. This confusion is caused by the shadow cast by the external light source.

6.4 Future work

Future work on this project should explore the autonomous navigation of the robot. The robot should be able to navigate through the farm while actively avoiding obstacles. An autonomous navigation system would significantly improve this system because this robot moves with a fixed routine. Hence, it is very susceptible to being derailed by obstacles. Since the tomato crops are planted in rows, a lane detection system would streamline the robot's navigation through the farm.

When diseases are spotted, this system communicates exact GPS coordinates to the user. A better, user-friendly way would be to share the location of the diseased leaf in rows or columns. Such a system would require actual demarcations to be made on the farm. Each row of crops would be given a specific number, and each row would have respective subdivisions to help identify areas on the farm quickly. The GPS coordinates of the entire farm would have to be categorized into these rows and columns. When this is done, instead of the farmer receiving

raw GPS coordinates as a location, the farmer would receive a column number and a row number that he is already familiar with.

Perhaps the most critical improvement that could be made to this system would be with the camera. An improvement with the camera would make the system functional in low light. A camera with better sensor sensitivity will capture features clearly, even in low light. An artificial light source can also be embedded into the robot. This way, it would be able to light the path for the camera. The model would also perform better with an artificial light source, as discussed in chapter 5.

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., & Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. *12th {USENIX} symposium on operating systems design and implementation*, 265-283.

Behmann, J., Mahlein, A. K., Rumpf, T., Römer, C., & Plümer, L. (2015). A review of advanced machine learning methods for the detection of biotic stress in precision crop protection. *Precision Agriculture*, 239-260.

Bradley, D., & Roth, G. (2007). Adaptive thresholding using the integral image. *Journal of graphics tools*, 13-21.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*(6), 679-698.

Dube, J., G., D., & Maphosa, M. (2020). Tomato Breeding In Sub-Saharan Africa - Challenges and Opportunities: A review. *African Crop Science Journal, Vol. 28*, 131 - 140.

Gullino, M. L., Albajes, R., & Nicot, P. C. (2020). *Integrated pest and disease management in greenhouse crops.* Switzerland: Springer.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv*.

Islam, S., & Digneffe, L. (2012). *Investing in Africa: The challenge of agriculture.* Belgium: Friends Of Europe.

Ivannikov, V., Morozov, S., Semenov, V., Tarlapan, O., Rasche, R., & Jung, T. (2000). Parallel object-oriented modeling and visualization in OpenMV environment. *In Proceedings of GraphiCon, 99*, 206-213.

Jankowski, S., Covello, J., Bellini, H., Ritchie, J., & Costa, D. (2014). *The Internet of Things: Making sense of the next mega-trend.* Goldman Sachs.

Jia, C., & Evans, B. L. (2012). Probabilistic 3-d motion estimation for rolling shutter video rectification from visual and inertial measurements. *In 2012 IEEE 14th International Workshop on Multimedia Signal Processing*, 203-208.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (n.d.). Gradient-based learning applied to document recognition. *Gradient-based learning applied to document recognition.*, 2278-2324.

Liu, Z., Guan, W., & Wen, S. (2019). Improved target signal source tracking and extraction method based on outdoor visible light communication using an improved particle filter algorithm based on Cam-Shift algorithm. *IEEE Photonics Journal, 11*(6), 1-20.

Louis, M. S., Azad, Z., Delshadtehrani, L., Gupta, S., Warden, P., Reddi, V. J., & Joshi, A. (2019). Towards deep learning using TensorFlow lite on RISC-v. *In Third Workshop on Computer Architecture Research with RISC-V (CARRV)*, 6.

Mahlein, A. K. (2016). Plant disease detection by imaging sensors–parallels and specific demands for precision agriculture and plant phenotyping. *Plant disease*, 241-251.

Martey, E., Al-Hassan, R. M., & Kuwornu, J. K. (2012). Commercialization of smallholder agriculture in Ghana: A Tobit regression analysis. *African Journal of Agricultural Research, 7*(14), 2131-2141.

Martinez, B., Monton, M., Vilajosana, I., & Prades, J. D. (2015). The power of models: Modeling power consumption for IoT devices. *IEEE Sensors Journal*, 5777-5789.

Mavridou, E., Vrochidou, E., Papakostas, G. A., Pachidis, T., & Kaburlasos, V. G. (2019). Machine vision systems in precision agriculture for crop farming. *Journal of Imaging*, 89.

Melomey, L. D., Danquah, A., Offei, S. K., Ofori, K., Danquah, E., & Osei, M. (2019). Review on tomato (Solanum Lycopersicum, L.) improvement programmes in Ghana. *Recent advances in tomato breeding and production*(59), 2-4.

Mizubuti, E. S., & Fry, W. E. (2006). Potato late blight. *The epidemiology of plant diseases*, 445-471.

Murakami, M., & Honda, N. (1996). An exposure control system of video cameras based on fuzzy logic using color information. *Proceedings of IEEE 5th International Fuzzy Systems*, 2181-2187.

Pallavi, S., Mallapur, J. D., & Bendigeri, K. Y. (2017). Remote sensing and controlling of greenhouse agriculture parameters based on IoT. *In 2017 International Conference on Big Data, IoT and Data Science*, 44-48.

Quweider, M. K., Scargle, J. D., & Jackson, B. (2007). Grey level reduction for segmentation, thresholding, and binarisation of images based on optimal partitioning on an interval. *IET Image Processing, 1*(2), 103-111.

Reyns, P., Missotten, B., Ramon, H., & De Baerdemaeker, J. (2002). A review of combine sensors for precision farming. *Precision Agriculture*, 169-182.

Sanchez-Iborra, R., & Skarmeta, A. F. (2020). TinyML-Enabled Frugal Smart Objects: Challenges and Opportunities. *IEEE Circuits and Systems Magazine*, 4-18.

Sarfo, N. Y. (2018). *Importance, Source, and Control of Bacteria Wilt Disease in Greenhouse Tomato (Solanum Lycopersicum L.) in Southern Ghana.* Doctoral dissertation, University of Ghana.

Satria, D., Yana, S., Munadi, R., & Syahreza, S. (2017). Prototype of Google Maps-Based Flood Monitoring System Using Arduino and GSM Module. *Int. Res. J. Eng. Techno*, 1044-1047.

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data, 6*(1), 1-48.

Silva, C. B., Pinto, F. A., Müller, C. A., & Moura, A. D. (2007). The economic feasibility of precision agriculture in Mato Grosso do Sul State, Brazil: a case study. *Precision Agriculture*, 255-265.

Warden, P., & Situnayake, D. (2019). *Tinyml: Machine learning with TensorFlow lite on Arduino and ultra-low-power microcontrollers.* Sebastopol: O'Reilly Media, Inc.

Xie, X., Ma, Y., Liu, B., He, J., Li, S., & Wang, H. (2020). A deep-learning-based real-time detector for grape leaf diseases using improved convolutional neural networks. *Frontiers in plant science, 11*, 1.