

# DEVELOPMENT OF A DEEP LEARNING-BASED MOBILE APPLICATION TO DETECT FRESHNESS OF FRUITS

SHADIA TALAL SALEM ABO EYADA (S.N 0419140012)

A project Submitted in Partial Fulfillment of the Requirements for the Degree of Master of  
Engineering in Computer Science and Engineering



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
MILITARY INSTITUTE OF SCIENCE AND TECHNOLOGY  
DHAKA, BANGLADESH

JANUARY 2023

DEVELOPMENT OF A DEEP LEARNING-BASED MOBILE APPLICATION TO  
DETECT FRESHNESS OF FRUITS

M. Engineering Thesis

By

SHADIA T S ABO EYADA (S.N 0419140012)

Approved as to style and content by the Board of Examination on 17 January 2023

---

Dr.Nusrat Sharmin  
(Supervisor)

Chairman

Assistant Professor of Computer Science and Engineering  
Examination  
MIST, Dhaka

Board of

---

Dr. Mohammad Nurul Huda

Member (External)

Professor of Computer Science and Engineering  
Examination  
UIU, Dhaka

Board of

---

Dr. Md. Mahbubur Rahman

Member (Internal)

Professor of Computer Science and Engineering  
Examination  
MIST, Dhaka

Board of

Senior Instructor of Computer Science and Engineering  
officio)  
MIST, Dhaka

Member (Ex-

# DEVELOPMENT OF A DEEP LEARNING-BASED MOBILE APPLICATION TO DETECT FRESHNESS OF FRUITS

## DECLARATION

I hereby declare that the study reported in this project entitled above is my own original work and has not been submitted anywhere for any degree or other purposes. Further, I certify that the intellectual content of this project is the product of my own work and that all the assistance received in preparing this project and sources have been acknowledged and/or cited in the reference section.

---

Shadia Talal Salem Abo Eyada  
Student No: 0419140012

Department of Computer Science and Engineering, MIST, Dhaka

## **ABSTRACT**

### **Development of a Deep Learning-Based Mobile Application to Detect Freshness of Fruits**

Fruits play a crucial role in our diet as they contain nutrients that are vital for our health. These nutrients are important as they help us protect against chronic diseases. Fruits that are not fresh will not contain as many nutrients than when it was fresh. Thus, it is important to ensure that only fresh fruits are consumed. However, there exist a large number of consumers that do not know how to select fruits that are fresh when purchasing. Besides that, the fruit industry uses harmful chemicals in order to perform fruit inspections which makes the fruit to lose its nutrients. To solve the problems stated above, this paper proposes a mobile application that can detect freshness in fruits. To do this, this project utilizes Deep Learning technologies in conjunction with a mobile application in order to predict the freshness of fruits in real time. Although there are several applications that can perform fruit freshness prediction, they require the user to have several external devices in order to accurately predict its freshness. Therefore, this project focused on developing an application that can do fruits freshness prediction in real time without needing extra devices.

## **ACKNOWLEDGMENT**

First and foremost, I want to give thanks and praise to God, the Almighty, for His blessings that helped me finish the job successfully.

I would like to express my deep and sincere gratitude to my supervisor, Dr. Nusrat Sharmin, Assistant Professor of Computer Science and Engineering, MIST, Dhaka, for providing invaluable guidance throughout this project. She has inspired me with her vision, integrity, and drive. I must also confess that she kept me striving hard to accrue the maximum out of this project to improve my academic knowledge in a more professional and aptly manner. It is because of her that I was able to complete my work not only in a timely but in a comprehensive way.

I would like to thank the Department of Computer Science and Engineering (CSE) of the Military Institute of Science and Technology (MIST), Mirpur-12, Dhaka, Bangladesh for supporting this project.

I want to express my sincere gratitude to my family and friends for their unwavering support and patience as I worked on my project. Last but not least, I want to express my gratitude to everyone who helped me, directly or indirectly, finish the work.

## TABLE OF CONTENTS

DECLARATION .....	ii
ABSTRACT.....	iii
ACKNOWLEDGMENT .....	iv
TABLE OF CONTENTS.....	v
LIST OF FIGURES.....	viii
LIST OF TABLES .....	x
CHAPTER 1 .....	1
INTRODUCTION .....	
1.1 Research Background.....	1
1.2 Motivation.....	1
1.3 Objectives and Outcomes.....	2
1.4 Main Contributions.....	2
1.5 Organization of the Remaining Chapters.....	3
CHAPTER 2 .....	4
THEORITICAL BACKGROUND AND LITERATURE REVIEW .....	
2.1 Theoretical Background.....	4
2.1.1 Image Processing.....	4
2.1.2 Artificial Intelligence.....	5
2.1.3 Deep Learning.....	5
2.2 Existing Systems.....	5
2.2.1 Desktop Based Studies.....	6
2.2.2 Mobile Application-based Fruit Freshness Detection.....	8
2.3 Research Gap.....	10

CHAPTER 3	11
METHODOLOGY	
3.1 Requirement Analysis.....	11
3.1.1 Performance Requirements.....	11
3.1.2 Functional Requirements.....	11
3.1.3 Non-Functional Requirements.....	12
3.2 Feasibility Study.....	12
3.3 Design.....	12
3.3.1 Iterative Development Approach.....	12
3.3.2 Work Flow Diagram.....	13
3.3.3 ER Diagram.....	14
3.3.4 Use Case Diagram.....	15
3.3.5 Sequence Diagram.....	15
3.4 Implementation.....	16
3.4.1 Proposed Model.....	17
3.4.2 Model Training.....	18
3.4.3 Model Evaluation.....	19
3.4.4 Tools and Language to Use.....	19
CHAPTER 4	21
EXPERIMENTSANDRESULTS	
4.1 Experiment Setup.....	21
4.2 Fruit Dataset Preparation.....	21
4.2.1 Dataset.....	21
4.2.2 Data Augmentation.....	22

4.3	Application Layout.....	23
4.4	Evaluation on Real Data.....	27
4.4.1	Valid Fruit Test Cases.....	27
4.4.2	Invalid Fruit Test Cases.....	28
CHAPTER 5		34
TESTING AND EVALUATION		
5.1	Functional Testing .....	34
5.1.1	Black Box Testing.....	34
5.1.2	White Box Testing.....	35
5.1.3	Performance Testing.....	35
5.1.4	Data Testing.....	36
CHAPTER 6		39
CONCLUSION AND MAIN OUTCOMES		
6.1	Project Review.....	39
6.2	Novelties and Contributions.....	39
6.2.1	Contributions.....	39
6.3	Limitations.....	40
6.4	Future Work.....	40
	Bibliography.....	41



## LIST OF FIGURES

Figure 3.1	Iterative Development Approach.....	13
Figure 3.2	Work Flow Diagram.....	13
Figure 3.3	ER Diagram.....	14
Figure 3.4	Use Case Diagram of the Application.....	15
Figure 3.5	Sequence Diagram for Fruit Inspection.....	16
Figure 3.6	Overall Architecture of the Proposed Method.....	17
Figure 3.7	Architecture of the proposed Model.....	18
Figure 3.8	Model Accuracy and Loss Graphs.....	19
Figure 4.1	Overall Design Steps.....	22
Figure 4.2	Dataset (sample).....	22
Figure 4.3	Dataset (sample).....	23
Figure 4.4	Data Augmentation.....	24
Figure 4.5	Application Main Pages.....	25
Figure 4.6	Valid Fresh Cases Test.....	25
Figure 4.7	Valid Not Fresh Cases Test.....	26
Figure 4.8	Invalid Cases Test.....	26
Figure 4.9	Fruit Information Pages.....	27
Figure 4.10	Fruit Information Pages (continue).....	27
Figure 4.11	Fruit Information Pages (continue).....	29
Figure 4.12	Valid Test Cases.....	30
Figure 4.13	Valid Test Cases (Continue).....	31
Figure 4.14	Valid Test Cases.....	32
Figure 4.15	Invalid Fruit Cases.....	33
Figure 4.16	Invalid Fruit Cases.....	34

Figure 5.1	Questionnaire Sample.....	36
Figure 5.2	Questionnaire Result Sample.....	37

## LIST OF TABLES

Table 2.1	Requirement Analysis Comparison.....	9
Table 2.2	Features Analysis Comparison.....	9
Table 5.2	Data Testing.....	38

# CHAPTER 1

## INTRODUCTION

### 1.1 Research Background

Fruits are a crucial component of a healthy diet, they provide many essential nutrients such as vitamins, minerals, and fiber. They are low in calories and can help boost the immune system, improve cardiovascular health, and prevent various diseases. Some of the specific benefits of consuming fresh fruits include reduced risk of heart disease and certain cancers, as well as antioxidants and other compounds that promote overall health and well-being. While fresh fruits are delicious and can be eaten as a snack or included in meals and beverages, it's important to make sure they are fresh to avoid any potential negative effects on taste and health. Determining the freshness of fruits can be difficult, as it often relies on subjective factors such as appearance and texture.

For individuals who are inexperienced in grocery shopping, it can be challenging to determine the freshness of fruits before purchasing them. It takes a significant amount of experience to select fresh fruits from the various options available in markets. While some people possess the necessary knowledge and skills, others may struggle with this task. Therefore, it would be beneficial to have an application that can accurately assess the freshness of fruits before they are bought. Such a tool has the potential to transform the way we shop for and consume fruits. It can enable consumers to make informed decisions about the quality of the fruits they purchase and aid fruit sellers and suppliers in better managing their inventory and reducing food waste.

### 1.2 Motivation

Health is one the most vital aspects that are of greater concern to most human beings at present days. To keep people healthier, they intake a lot of fruits which contain more amount of nutrition and help them to remain fit. The Nutrients provided by fruits are vital for the proper maintenance of the body. But those fruits nowadays are ripened through some artificial means like the usage of chemicals like calcium carbide as the ripening agent which might even cause cancer. The motivation for developing an application to detect fresh fruits is to improve quality, efficiency, sustainability, safety, and cost-effectiveness in the fruit industry.

### **1.3 Objectives and Outcomes**

The main objective of this project is to develop a deep learning-based mobile application that can detect the freshness of twelve types of fruits (apple, banana, mango, orange, watermelon, pear, strawberry, peach, pineapple, blueberry, grape, pomegranate) in real time. To achieve this objective, the following sub-objectives are considered:

- Research and review the state of the art in deep learning and computer vision techniques for fruit freshness detection. This involved studying existing approaches and identifying the most promising methods for our purpose.
- Implement a deep learning model for fruit freshness detection. This involved selecting appropriate data sets, designing the model architecture, and training and fine-tuning the model using appropriate techniques.
- Integrate the model into a mobile application that allows users to easily determine the freshness of fruits. This involved designing an intuitive user interface and integrating the model into the app.
- Test and evaluate the performance of the developed mobile application. This involved conducting experiments to assess the accuracy and reliability of the app in detecting the freshness of different types of fruits.
- Document the development process and results in a final report. This involved describing the methods used, the results obtained, and any limitations or future work

In short, the objectives are as follows:

- To design and develop a deep learning-based mobile application to recognize fresh fruits and display nutritious values in real-time.
- To evaluate the functionality of the developed application.

The outcomes of the project are:

- A deep learning-based mobile application on smartphones as a real-time app to recognize fresh fruit along with its nutrition.

### **1.4 Main Contributions**

The main contribution of this thesis has been listed below

- A deep learning-based mobile application that recognizes fresh fruits in real-time.
- A deep learning-based mobile application that gives the nutrition information of each type.

## 1.5 Organization of the Remaining Chapters

The introductory chapter has presented the basic introduction of the aim of the recognition system, the motivation, and the objectives of the project, the following chapters attempt to convey further details and approach to developing the fruit freshness model. The report further developed in the following sequences:

- *Chapter 2 Background and Literature Review* will describe details of the theoretical background and Review of existing applications.
- *Chapter 3 Methodology* will present the design and implementation part.
- *Chapter 4 Experiment and Results* will present the layout of the developed app and also the experiment results.
- *Chapter 5 Testing and Evaluation* presents the test cases for different types of fruit inspection.
- *Conclusion* will conclude the review of the project by summarizing that.

# CHAPTER 2

## THEORITICAL BACKGROUND AND LITERATURE REVIEW

Previous research has mainly focused on a limited range of fruits, including apples, bananas, and peaches, and it is unclear to what extent these findings can be generalized to other types of fruit. Additionally, previous studies have mostly examined the use of image processing in controlled laboratory environments, and it is uncertain how well these findings would translate to real-world situations, such as during storage or transportation. While image processing has shown promise in prior research, its performance in real-world applications and potential limitations are currently unknown. By addressing these and other potential gaps in the current literature, this project aims to contribute to a better understanding of the potential and limitations of deep learning for detecting the freshness of the fruit. In this chapter, we aim to discuss the theoretical background and literature review.

### 2.1 Theoretical Background

#### 2.1.1 Image Processing

Image processing is a set of techniques for analyzing and manipulating digital images (Haenlein and Kaplan 2019). It involves the use of computer algorithms to perform operations on images in order to extract useful information or to enhance the image in some way. There are many different techniques that can be used in image processing, including image filtering, image enhancement, image restoration, and image analysis (Kattenborn et al. 2021). Processing images requires different techniques, including:

- *Filtering*: This involves applying a function to each pixel in an image to enhance or suppress certain features. Filtering techniques include smoothing filters, which reduce noise and detail in an image and sharpening filters, which enhance the edges and details in an image.
- *Segmentation*: This involves dividing an image into regions or segments, each of which corresponds to a different object or background. Segmentation is often used as a preprocessing step for object recognition or to separate different objects in an image for further analysis.
- *Feature extraction*: This involves extracting important or distinctive characteristics of an image or object, which can be used for classification or recognition. Features might include edges, corners, or other image structures.

- *Classification*: This involves assigning labels or categories to objects in an image based on their characteristics. Classification is often used in combination with feature extraction to identify objects or classify images into predefined categories.
- *Reconstruction*: This involves using image processing techniques to restore or improve the quality of an image, such as removing noise or restoring detail.
- *Registration*: This involves aligning or aligning multiple images or video frames to a common reference frame, which is often necessary for tasks such as image-stitching or object tracking.

## 2.1.2 Artificial Intelligence

Artificial intelligence (AI) is a broad field that encompasses the development of computer systems that can perform tasks that normally require human intelligence, such as learning, problem-solving, decision-making, and pattern recognition (Haenlein and Kaplan 2019). AI systems can be trained to perform a wide variety of tasks, ranging from simple tasks like language translation to more complex tasks like self-driving cars or medical diagnosis. There are several sub fields of AI, including machine learning, which involves training computer systems to learn from data, and natural language processing, which involves enabling computers to understand and generate human-like language.

## 2.1.3 Deep Learning

Deep learning is a subfield of machine learning that is inspired by the structure and function of the brain, specifically the neural networks that make up the brain (Koirala et al. 2019). It involves training artificial neural networks on a large data set, allowing the network to learn and make intelligent decisions on its own (Sa et al. 2016). Deep learning techniques have been particularly successful in tasks that involve large amounts of data and complex patterns, such as image and speech recognition, natural language processing, and machine translation (Roy, Chaudhuri, and Pramanik 2021). Deep learning algorithms are often implemented using neural networks, which are composed of layers of interconnected nodes. Each node processes input data and passes it on to the next layer, with the final layer producing the output. The layers in between the input and output layers are called hidden layers, and the process of training a deep learning model involves adjusting the weights and biases of the connections between the nodes in order to optimize the output.

### 2.1.3.1 Convolutional neural network (CNN)

A convolutional neural network (CNN) is a type of artificial neural network that is designed to process data from multiple sources and predict outcomes based on that data (Alzubaidi et al. 2021). It is particularly well-suited to image and video recognition tasks, as it is able to analyze the spatial relationships between pixels in an image or



frame and use that information to classify the image or video. CNNs are composed of layers of interconnected nodes, each of which performs a specific operation on the input data and passes it on to the next layer (Kattenborn et al. 2021). Convolutional neural networks (CNNs) are composed of several different types of layers, each of which performs a specific function:

- **Input Layer:** This is the first layer in the network, and it receives the raw input data (e.g. an image).
- **Convolutional Layer:** This layer applies a convolution operation to the input data, using a set of learnable filters (also called kernels or weights). The filters are used to detect specific features or patterns in the input data.
- **Pooling Layer:** This layer down samples the output of the convolutional layer by applying a pooling operation, such as max pooling or average pooling. The pooling operation reduces the dimensionality of the data and helps to prevent overfitting.
- **Fully Connected Layer:** This layer combines the features extracted by the convolutional and pooling layers and makes a final prediction or classification. The fully connected layer is composed of a set of nodes, each of which is connected to every node in the previous layer.
- **Output Layer:** This is the final layer in the network, and it produces the final prediction or classification based on the output of the fully connected layer.

## **2.2 Existing Systems**

The literature review is organized into two sections. The first section comprises a review of comparable applications and a comparison of these applications to the current one. The second section consists of a review of research conducted on desktop-based systems.

### **2.2.1 Desktop Based Studies**

#### **2.2.1.1 Machine Learning Method**

- (Mythri et al. 2020) developed an artificial neural network (ANN) model to recognize the ripeness of bananas using a histogram approach. The study began by collecting three sets of images of unripe, ripe, and overripe bananas, which were captured using a webcam and then resized to 352x288. The researchers then extracted the RGB components of the images and plotted a histogram based on the RGB values. The ANN model was trained using a three-stage propagation process that involved the feed-forward of input training patterns, the back-propagation of errors, and weight adjustment. The output of the model was compared to the target value to calculate the appropriate error, which was then distributed back to the hidden layer. After the network was

trained, it was used to classify the ripeness of bananas through a feed-forward phase. A gradient descent method was used to adjust the weights to minimize the output's squared error. The researchers also created a graphical user interface (GUI) so that users could easily employ the ANN model to detect the ripeness of bananas. While the model was able to correctly classify 25 out of 28 samples, the authors identified several areas for improvement, such as using a higher resolution camera, removing unnecessary background components, and increasing the number of color intensity groups.

- (Mazen and Nashat 2019) developed a method for classifying the ripeness of bananas using an artificial neural network (ANN). To create their model, the authors first collected a dataset of 300 images of bananas with varying levels of ripeness, which were captured using a Samsung Note 3 camera. The images were then pre-processed, including using an HSV model to describe the RGB colors and applying morphological filtering and Otsu's method to remove the background. The model calculated the ripeness factor of the bananas based on the number of brown spots in the images and used statistical texture analysis to determine the ripeness. The ANN was trained using the Levenberg-Marquardt backpropagation algorithm and consisted of an input and output layer with 4 neurons and 10 hidden layers. The dataset was partitioned into a 70:30 split for training and testing, with the goal of evaluating the accuracy and precision of the model.
- (Salah and Hassan 2016) reviewed the various machine-learning techniques that have been applied to fruit detection and classification. They discussed the advantages and disadvantages of different methods, such as decision tree algorithms, artificial neural networks, support vector machines, and more. They also discuss the different types of features used in fruit detection, such as color, texture, and shape.
- (Gunathilake, Perera, and Jayasekara 2018) discussed the use of machine learning for fruit detection and classification, focusing on the use of deep learning techniques. The authors reviewed the various deep learning architectures that have been applied to fruit detection, such as convolutional neural networks and multi-layer perceptrons. They also discussed the use of different types of features, such as color, texture, and shape, and the challenges of fruit detection in different environments.

#### **2.2.1.2 Deep Learning Methods**

- (Dhiman, Y. Kumar, and M. Kumar 2022) proposed a method for detecting defective apples using a computer vision system combined with deep learning methods. The system involves a 4-lane sorting system that utilizes a conveyor belt to move and rotate the apples so that they can be captured in multiple orientations by two cameras. The images are then captured using linear lights to ensure sufficient brightness. To further improve accuracy, the system is placed inside a light chamber to block out stray light. The authors did not mention the specific deep learning methods used in the system.
- (Pande et al. 2019) proposed an efficient approach for classifying and grading fruits using a deep convolutional neural network (CNN). Their method involves an automated system in which fruits are placed on a conveyor

belt and images are captured by a camera. The images are then pre-processed using a Raspberry PI and fed into a DCNN classifier, which runs on a graphics processing unit (GPU). The authors did not provide further details on the specific methods used for pre-processing the images or training and evaluating the DCNN classifier.

- (Murthy, Prasad, and Krishna 2019) presented a deep learning-based method for detecting the freshness of apples using convolutional neural networks (CNNs). They collected a dataset of images of fresh and rotten apples and trained a CNN on this dataset. The CNN was able to achieve an accuracy of 95
- (Karim, Imran, and Rahman 2019). In this paper, the authors presented a method for detecting the freshness of bananas using deep learning. They collected a dataset of images of fresh and rotten bananas and used transfer learning to fine-tune a pre-trained CNN on this dataset. The fine-tuned CNN was able to achieve an accuracy of 95
- (Islam and Imran 2019). In this paper, the authors presented a method for detecting the freshness of apples using deep learning. They collected a dataset of images of fresh and rotten apples and trained a CNN on this dataset. The CNN was able to achieve an accuracy of 95

### 2.2.2 Mobile Application-based Fruit Freshness Detection

There are only a limited number of applications that are capable of detecting the freshness of fruits that are readily available online. Four of these applications were selected for review. These applications use various techniques, such as image processing, machine learning, and deep learning to assess the freshness of fruits based on factors such as color, texture, and shape. While previous research has demonstrated the potential of image processing for detecting the freshness of fruits, it is not clear how well these techniques would perform in real-world applications or what the limitations of this approach might be.

- *Clari Fruit*: An application designed for detecting the freshness of fruits. It was developed in 2018 and is currently in the pre-release beta stage. Users who are interested in accessing the application can request to join the beta program, which will provide them with access to the tool. Clari Fruit can be downloaded from the AppStore and PlayStore for Android devices. The primary purpose of this application is to perform quality control and provide users with a data analytics platform for fresh produce. It is intended to help users make informed decisions about the quality of the fruits they purchase and to improve the efficiency, sustainability, and cost-effectiveness of the fruit industry.
- *Fruit and Veg Detector*: A freshness detection application developed by Dragster Production and available on the Google Play Store. It was created in Pakistan and last updated in August 2020. The application has been downloaded over 5,000 times and has a rating of 4.6 stars. It is a user-friendly tool that allows users to quickly assess the freshness of fruits by simply pointing their camera at the desired fruit. The application

was developed with the aim of providing users with an easy and convenient way to determine the freshness of produce. It is designed to be simple and straightforward to use, requiring minimal effort on the part of the user. Fruit and Veg Detector has the potential to be a useful resource for individuals seeking to make informed decisions about the quality of the fruits they purchase.

- *FreshCheck*: This application allows users to quickly assess the freshness of fruits by scanning them with their smartphone camera. It uses image processing technology to analyze the color, texture, and shape of the fruit in order to determine its freshness. FreshCheck also provides users with information about the nutritional content of the fruit and offers storage and handling tips to help preserve its freshness.
- *FreshnessPro*: This application uses a combination of machine learning algorithms and sensory analysis to determine the freshness of fruits. It allows users to take a photo of the fruit and receive a freshness score, as well as recommendations for optimal storage and handling. It also provides users with information about the nutritional content of the fruit and offers recipe ideas for using it in meals.

### 2.2.2.1 Comparative Analysis

Freshness detection applications have the potential to be useful tools for determining the freshness of fruits and making informed decisions about the quality of produce. However, there are also several limitations to consider when using these types of applications.

Table 2.1: Requirement Analysis Comparison

<i>Requirements</i>	<i>Application</i>				
	Clari Fruit	Fruit Veg Detector	FreshCheck	FreshnessPro	<b>DFruit</b>
<i>Camera</i>	Yes	Yes	Yes	Yes	Yes
<i>Hardware Requirements</i>	High	Low	High	High	High
<i>Internet Access</i>	No	No	No	No	No

Table 2.2: Features Analysis Comparison

<i>FEATURES</i>	<i>APPLICATION</i>				
	Clari Fruit	Fruit Veg Detector	FreshCheck	FreshnessPro	<b>DFruit</b>
<i>Real Time</i>	No	Yes	No	No	Yes
<i>Simplicity</i>	Moderate	Easy	Complex	Complex	Easy
<i>Give Fruit Details</i>	No	No	Yes	Yes	Yes
<i>Freshness Label</i>	Yes	No	No	No	Yes
<i>Detect 10 Fruit Plus</i>	No	Yes	Yes	No	Yes

Neither ClariFruit nor Fruit and Veg Detector can automatically identify the type of fruit before inspecting its freshness (see Table 2.1). FreshCheck and FreshnessPro also lack the ability to detect the freshness of the fruit in real time 2.2. The final product of this project addresses these gaps by including a feature that can identify the type of fruit and detect its freshness in real-time. The design of the final product was informed by the analysis of existing systems and the identification of areas for improvement, with the aim of maximizing its potential to meet the needs of users.

## 2.3 Research Gap

The main goal of this project is to create a mobile application that can help users, particularly those who lack experience, select fresh fruits. In order to achieve this goal, existing systems were reviewed to identify gaps. Neither ClariFruit nor Fruit and Veg Detector has the ability to automatically detect the type of fruit before performing a freshness inspection. The final product of this project address this gap by including a feature that can identify the type of fruit before inspecting its freshness. Neither FreshCheck nor FreshnessPro has the ability to detect the freshness of the fruit in real time, but the final product of this project is able to detect the fruit in real time. By analyzing existing systems and applications areas for improvement were identified and incorporated into the design of the final product, ensuring that it has the maximum potential to meet the needs of users.

According to reviewed litterateurs, there are a few potential research gaps in the field of using deep learning techniques for detecting the freshness of fruits:

- *Generalizability*: Many previous studies have focused on detecting the freshness of specific types of fruit, such as apples or bananas. It is not clear how well the findings of these studies would generalize to other types of fruit. A mobile application that is able to detect the freshness of a wide range of fruit types would be useful in addressing this gap.
- *Real-world conditions*: Many previous studies have focused on detecting the freshness of the fruit in controlled laboratory settings. It is not clear how well the findings of these studies would generalize to real-world conditions, such as in storage or transportation. A mobile application that is able to detect the freshness of fruit under real-world conditions would be useful in addressing this gap.
- *Limitations of image processing*: Some previous studies have used image processing techniques to detect the freshness of the fruit. However, it is not clear how well these techniques would perform in real-world applications or what the limitations of this approach might be. A mobile application that is able to detect the freshness of fruit using a different approach, such as sensor data or chemical analysis, would be useful in addressing this gap.

Overall, a mobile application that is able to detect the freshness of a wide range of fruit types under real-world conditions using a variety of approaches would be useful in addressing these research gaps and improving the quality, efficiency, sustainability, safety, and cost-effectiveness of the fruit industry.

# **CHAPTER 3**

## **METHODOLOGY**

The chapter is organized into four main sections: requirements analysis, feasibility study, system design, and implementation. Requirements analysis involves defining, documenting, and organizing the requirements for a software system, taking into consideration the needs and goals of the stakeholders. A feasibility study evaluates the technical and financial viability of a proposed software project. System design involves developing the overall structure and technology of the system, as well as determining functional requirements. Implementation involves the creation and deployment of the system, including the selection of tools and testing of components.

### **3.1 Requirement Analysis**

The system being developed must meet the requirements outlined in the specification, which can be divided into two categories:

#### **3.1.1 Performance Requirements:**

- The system should be designed with database independence in mind.
- The system should have a fast response time.
- The system should have high throughput capabilities.

#### **3.1.2 Functional Requirements:**

- As a user, I should be able to access my camera to detect freshness of fruits.
- As a user, I should be able to see the result of inspection immediately.
- As a user, I should be informed with the type of fruits that can perform inspection.
- As a user, I want all the buttons in the application to be working.

### **3.1.3 Non-Functional Requirements:**

- As a user, I should be able to run the application on any Android device.
- As a user, I want the fruit freshness prediction to be 85 percent accurate of the time.
- As a user, I should be able to use the application without crashing.

## **3.2 Feasibility Study**

Feasibility studies have been conducted in the following areas:

- *Economic Feasibility:* The assessment of whether a proposed project will be financially feasible or not. It involves comparing the projected costs of the project to the expected benefits and savings. If the benefits are expected to outweigh the costs, the project is considered economically viable and is likely to be undertaken.
- *Technical Feasibility:* The assessment of whether a proposed project is technically feasible, or whether it can be completed with the resources and technology currently available. This includes evaluating the current hardware, software, and other technical resources of the organization to determine if they are sufficient to support the project.
- *Operational Feasibility:* The assessment of whether a proposed project is operationally feasible, or whether it can be completed and integrated into the current business operations successfully. This includes evaluating the impact of the project on the organization, such as the level of disruption to current operations, the level of resources required, and the level of change management that will be required.
- *Time Feasibility:* The assessment of whether a proposed project can be completed within the allocated time frame.

Based on the information gathered, the system is determined to be feasible in all areas.

## **3.3 Design**

### **3.3.1 Iterative Development Approach**

An iterative development approach was used for the creation of the mobile application. This approach involves repeatedly exposing the system to user feedback and evolving it through a series of development cycles until a suitable final product is achieved 3.1.

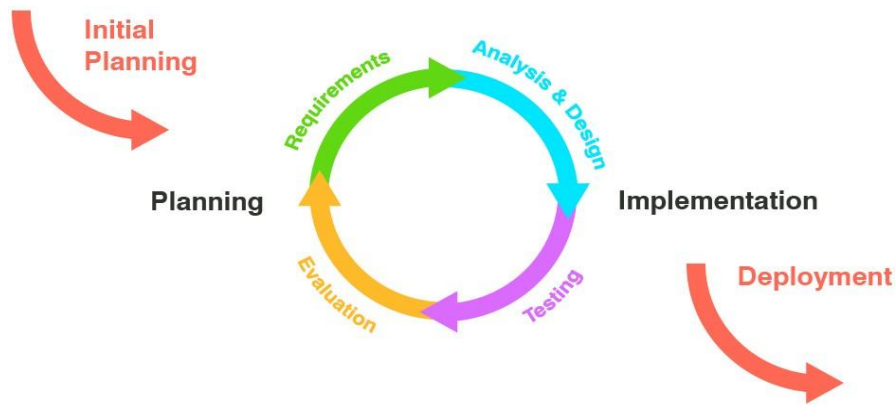


Figure 3.1: Iterative Development Approach

This approach is the most effective for the development of **Dfruit** due to the high level of user interaction with the application. By allowing users to test and provide feedback on an initial version of the application, valuable insights can be gathered and areas for improvement easily can be figured. This approach also allows for the rapid discovery and resolution of errors or glitches, ensuring that the final product is as user-friendly and reliable as possible.

### 3.3.2 Work Flow Diagram

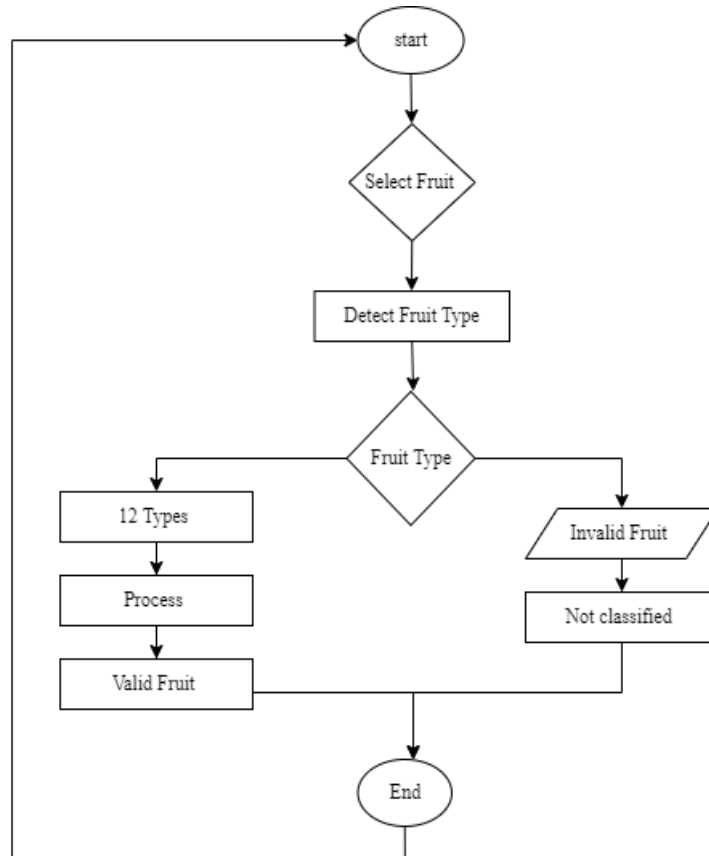


Figure 3.2: Work Flow Diagram



The mobile application is designed from scratch. Therefore, every page in the application needs to have elements that are arranged neatly so that the layout of the page appears neat and tidy. Although the design of the application may look simple, it still gets the job done and can be improved in the future. The application flow diagram is explained in figure 3.2. The diagram shows the overall flow of the application in detail. At the main page of the application, the user is able to proceed to the main page. After that, the user takes a photo using the camera. To point the camera the user is required to allow the application to access the camera application. Next, after pointing the camera the system will run the deep learning model to classify the type of fruit and then load in the freshness classification model according to the type of fruit detected. The results of the inspection will be shown to the user such as fresh or not fresh, type of fruit, and information about the fruit will be shown.

### 3.3.3 ER Diagram

An Entity-Relationship (ER) diagram is a graphical representation of the relationships between entities in a database. For this system case 3.3, the entities are fruits, which can be either valid or invalid. For valid fruits, they can be further classified as either fresh or not fresh. Similarly, for invalid fruits, they can also be classified as either fresh or not fresh. Each fruit will have a name, label, and nutrition information associated with it. In the ER diagram, the entity "Fruit" would have two attributes: "validity" and "freshness". The "validity" attribute would have two possible values: "valid" and "invalid," and the "freshness" attribute would also have two possible values: "fresh" and "not fresh." The relationship between the "Fruit" entity and the "Name," "Label," and "Information" entities would be one-to-one, as each fruit would have only one name, label, and set of nutrition information associated with it.

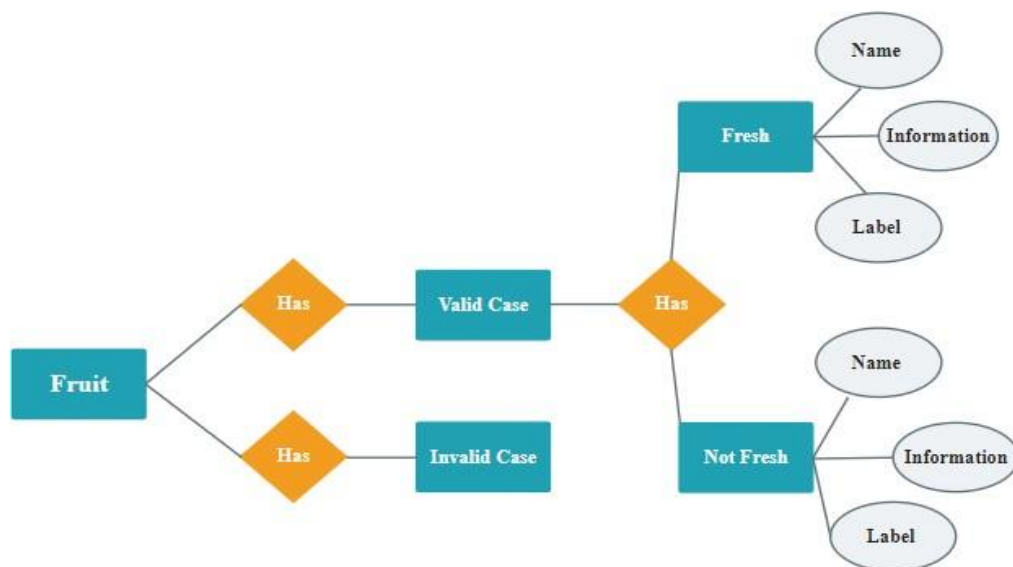


Figure 3.3: ER Diagram

### 3.3.4 Use Case Diagram

A use case diagram is a graphical representation of the interactions between a system and the entities that use it. The use case diagram 3.4 includes a single use case, called "Detect Freshness of Fruit," which represents the flow of events described in the scenario provided. This use case is initiated by a user who wants to determine the freshness of a fruit and is represented by an actor in the use case diagram. The flow of events for the "Detect Freshness of Fruit" use case is represented by a series of interactions between the system and the user:

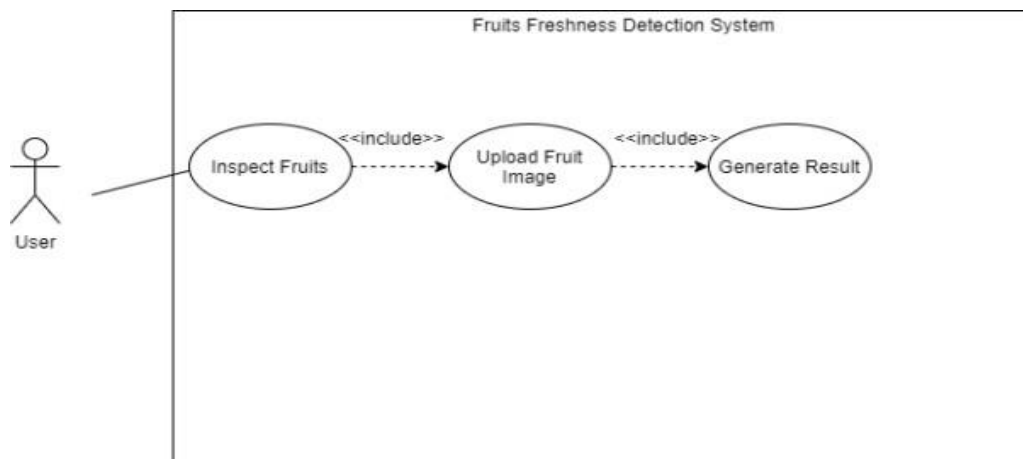


Figure 3.4: Use Case Diagram of the Application

- The first interaction in the flow of events would be the system prompting the user to upload an image of the fruit.
- The second interaction would be the user uploading the image of the fruit.
- The third interaction would be the system displaying the result of the inspection.

### 3.3.5 Sequence Diagram

A sequence diagram is a graphical representation of the interactions between objects in a system. It shows the order in which these interactions take place, as well as the messages that are exchanged between the objects. In the case of the freshness detection system, as shown in figure 3.5, the sequence diagram includes two objects: the user and the system. The system prompts the user to upload an image of the fruit. It will then utilize a fruit classification model to determine the type of fruit in the image. Once the fruit type has been identified, the system will load a deep learning model specifically for that type of fruit and use it to assess the freshness of the fruit. The system will generate a result based on this analysis, which will include the type of fruit and the freshness label (fresh or not fresh). Finally, the system will present the fruit information to the user in the form of a display. The sequence diagram for this system provides a detailed, step-by-step representation of the process of detecting the freshness of fruit, and the messages exchanged between the user and the system throughout the process (see Figure 3.5).

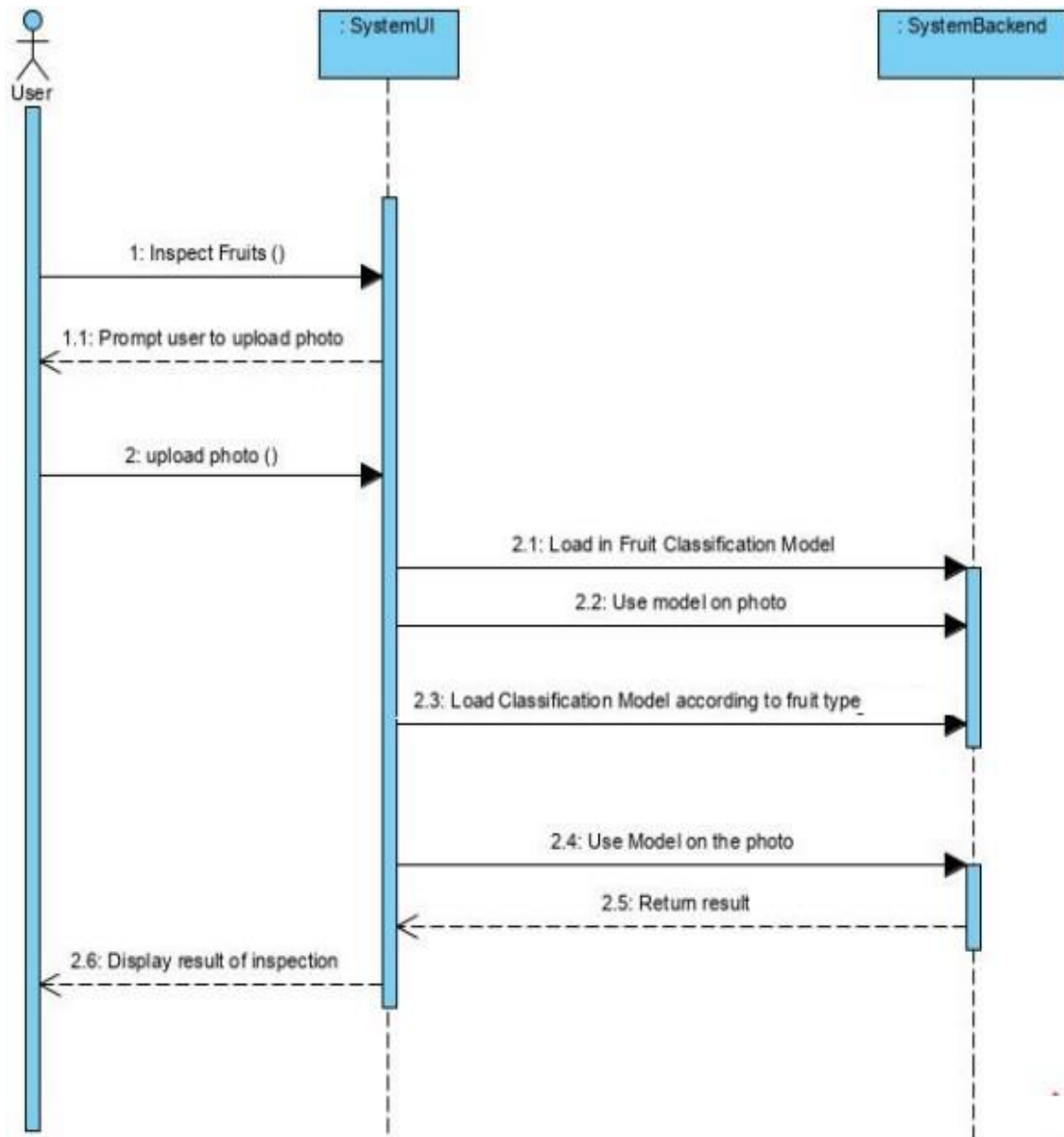


Figure 3.5: Sequence Diagram for Fruit Inspection

### 3.4 Implementation

The overall architecture of the proposed method is shown in Figure 3.6 . The three-phase methodology for developing the deep learning models involves collecting and preparing a data set, training a model on that data set, and then testing the trained model on a separate testing data set to evaluate its performance. In the data set phase, the data is cleaned and split into training and testing sets. In the model training phase, the deep learning model is presented with input data and learns to make predictions or decisions based on that data. In the model testing phase, the trained model is evaluated on the testing data set to determine how well it is able to generalize to new, unseen data.

The first step consisted of selecting the model to be trained to identify the type of fruit and its state, whether it was fresh or not, and then providing fruit-related health information based on the categorization. In the initial phase,

the Yolo V4 model was implemented. YOLOv4 (You Only Look Once) is a real-time object detection system, the model is able to detect objects with high precision, especially small objects. This makes it suitable for a wide range of applications such as surveillance, robotics, and autonomous vehicles Bochkovskiy, Wang, and Liao 2020.

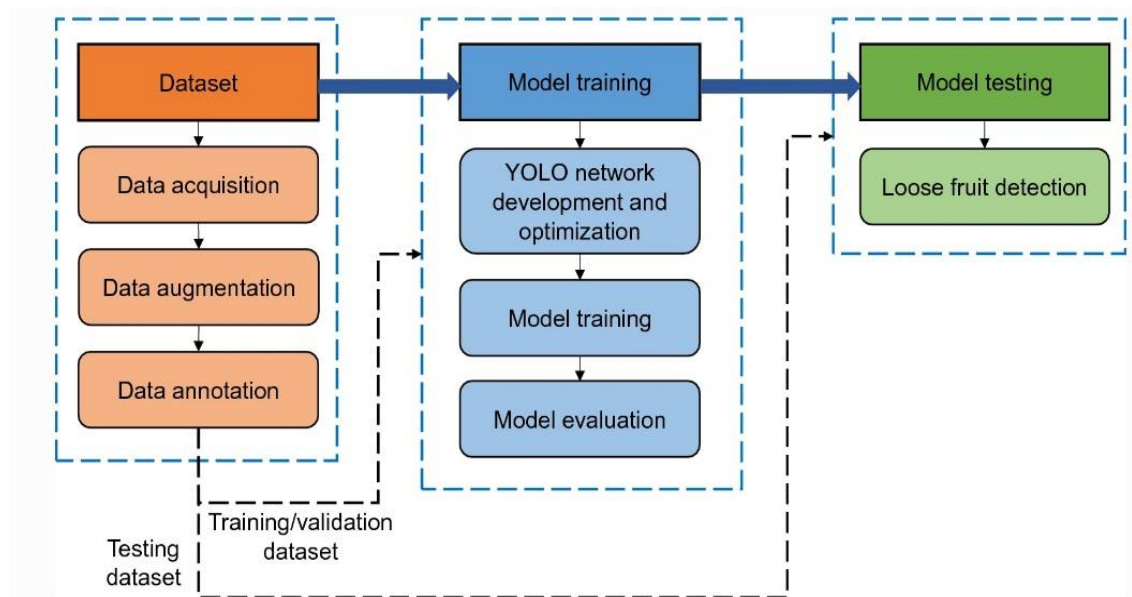


Figure 3.6: Overall Architecture of the Proposed Method

### 3.4.1 Proposed Model

Yolo v4 can process images and detect objects in real-time, making it suitable for use in applications where speed is critical. The model uses self-supervised pre-training and transfer learning to adapt to new environments and object classes. This makes it a versatile object detection system that can be used in a wide range of applications Gai, Chen, and Yuan 2021. YOLOv4 has a smaller model size compared to other state-of-the-art object detection models. This makes it easier to deploy and use in resource-constrained environments.

The model was trained using the acquired data, and the produced model was linked to the flutter UI kit to complete the system's development. However, because the libraries in flutter and the Yolo V4 model were incompatible, it was required to look for another model to train and link to flutter. Other versions of the same model were tested, and their compatibility with the mobile SSD was verified. After experimenting and testing the model, it turned out that the second version is compatible with flutter. Yolo v2 is more accurate, particularly in detecting small objects, and has a faster inference speed, making it suitable for real-time applications Zhao and Qu 2019. YOLOv2 is better at handling objects of different scales, it has a more efficient model, with fewer parameters and a smaller model size, making it easier to deploy and use in resource-constrained environments Lee 2021.

### 3.4.2 Model Training

Model training is the process of learning patterns in data and using those patterns to make predictions or decisions. In deep learning, a model is trained on a dataset, which is a collection of data used to teach the model to perform a specific task. During training, the model is presented with a series of input data and expected output data, and its internal parameters are adjusted in order to minimize the difference between the predicted output and the expected output. The goal of model training is to create a model that is able to make accurate predictions or decisions based on new, unseen data.

For the fruit freshness detection system YOLO v2 model is used to detect the fruit type and validity in real-time, by training the model on a dataset of images of fresh and spoiled fruit. The model would then be able to predict the freshness of a piece of fruit in a new image by analyzing its visual characteristics and comparing them to the characteristics of the fruit in the training dataset. The network used in this model is based on the YOLOv2 tiny architecture, with an additional fully connected layer added for classification of fruit freshness 3.7. This custom layer begins with a flattening layer, which converts the multi-dimensional tensor output of the YOLOv2 tiny layer into a single array. This is followed by a dense layer containing 1280 neurons, which receives the output of the final layer of YOLOv2 tiny as input. Tiny-yolo-dense is a modified version of Tiny-yolo that incorporates dense connectivity patterns. The dense block, which replaces the 7th convolutional layer, allows each layer to receive the feature maps from all previous layers and pass on its own feature map to all subsequent layers. This improves the flow of information and encourages the reuse and fusion of features at multiple levels, while also avoiding an increase in computational complexity.

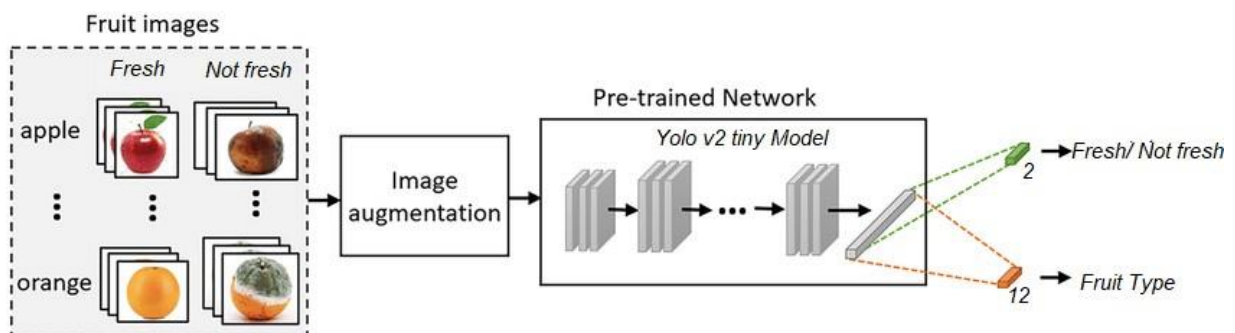


Figure 3.7: Architecture of the proposed Model

To address the challenge of detecting occluded or overlapped fruit, manually labeled samples of the foreground regions of these fruits were used to train a YOLOv2 model. This allowed the model to specifically focus on detecting the occluded or overlapped fruit rather than the background or other objects in the image. Extracting features only from the foreground region of an image, rather than the entire bounding box, can help to improve the learning of features specific to occluded or overlapped fruit. This is because it avoids extracting redundant features from

the non-target regions of the bounding box, allowing the model to focus on learning relevant features from the foreground region of the occluded or overlapped fruit. To address the issue of natural lighting variations impacting the appearance of fruit images, the first step taken was to apply adaptive histogram equalization to the training sample images to improve their quality and diversity of illumination. In addition to this, the foreground region and bounding box of the training samples were manually labeled. These steps were taken to improve the performance of the model.

### 3.4.3 Model Evaluation

The goal of this phase is to expose the version of the application developed to the users in order for them to evaluate it. After obtaining the feedback from users, this feedback is noted down and used to improve the application in the next iteration of the development cycle. The detection model processes incoming images of fruit, classifies them as "fresh" or "not fresh," and uses the Yolo algorithm and Darknet for training. The model is performing well during training, with a quite few drops indicating underfitting 3.8. The accuracy of the model is 81 percentage on both the training and test datasets is similar. The loss graphs for both the training and test data suggest that the model may be experiencing some underfitting. There are several notable dips in validation accuracy during the training process, specifically at around the 20th, 40th, and 80th epochs. These drops may be due to factors such as overfitting or the model encountering particularly challenging input images for classification in these epochs. Similar fluctuations can be seen in the loss graph, with the model experiencing higher loss values at roughly the same epochs as the dips in validation accuracy. These occurrences may warrant further investigation.

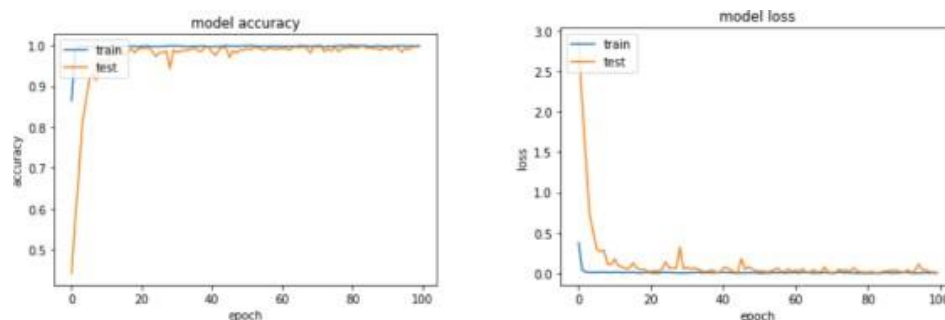


Figure 3.8: Model Accuracy and Loss Graphs

### 3.4.4 Tools and Language to Use

Programming Language

- Flutter Dart
- Python

## Software Requirements

- TensorFlow 1.15
- Python 3.7
- Flutter
- Visual Basic 6.0
- CUDA 11.2 toolkit
- Adobe Illustrator
- Google Drive
- Glob-Google
- Draw.io
- Visual Paradigm

## Hardware Requirements

- Laptop
- Mobile Device





















































	Simple		Real-world		Simple		Real-world	
Apple								
Banana								
Orange								
Pear								
Strawberry								
Others								

Figure 4.2: Dataset (sample)



Figure 4.3: Dataset (sample)

#### 4.2.2 Data Augmentation

Data augmentation is a technique used in machine learning to artificially increase the size of a dataset by generating new data samples from the existing data. This is often done in order to improve the performance of a model, particularly when the original dataset is small or limited in some way. In the case of deep learning, data augmentation can be particularly useful when training a model to recognize patterns in images, as it allows the model to be exposed to a wider variety of data. If the original dataset only includes images of fruit taken from a single angle, the model may have difficulty recognizing the fruit when it is presented from a different angle. Data augmentation can be used to generate additional images of the fruit from different angles, which can help the model to learn to recognize the fruit from any angle. If the original dataset is insufficient for training a deep learning model, data augmentation can be used to increase the size of the dataset and provide the model with additional data to learn from. This can help to improve the model's performance and allow it to generalize better to new, unseen data. Data augmentation is done

by using the function shown in Figure 4.4 .

```
[ ] def vert_horz(folder):
    folder_path = os.listdir(folder, num_augmation)
    count=0
    for img_name in folder_path:
        path_img = os.path.join(folder, img_name)
        img_read = cv2.imread(path_img)
        bright = np.ones(img_read.shape, dtype='uint8')*70
        bright_add = cv2.add(img_read, bright)

        #Flipping image vertically and horizontally
        vert_horz = cv2.flip(bright_add, -1)
        img_write = cv2.imwrite(os.path.join(folder,('AUG_'+img_name)), vert_horz)
        count+=1
        if count==num_augmation:
            break

[ ] def vertically(folder):
    folder_path = os.listdir(folder, num_augmation)
    count=0
    for img_name in folder_path:
        path_img = os.path.join(folder, img_name)
        img_read = cv2.imread(path_img)
        bright = np.ones(img_read.shape, dtype='uint8')*70
        bright_add = cv2.add(img_read, bright)

        #Flipping image vertically
        vert = cv2.flip(bright_add, 0)
        img_write = cv2.imwrite(os.path.join(folder,('AUG_'+img_name)), vert)
        count+=1
        if count==num_augmation:
            break
```

Figure 4.4: Data Augmentation

For the developed system case, after organizing the fruit images into different folders, the next step was to use data augmentation to increase the size of the dataset. Data augmentation is a technique that creates additional data by altering existing images in the dataset in various ways, such as rotating or flipping them. This can significantly increase the number of data points in the dataset, particularly when the original dataset is small. There are many types of data augmentation techniques, including rotation, flipping, zooming, and shifting, among others. Using data augmentation can significantly improve the accuracy of the model and prevent over-fitting. After performing data augmentation, the size of the dataset should be increased tenfold for each class.

### 4.3 Application Layout

The application only contains three pages (see Figure 4.5, the main page, the inspection page, and the information page. The main page only serves the purpose of redirecting the user to the inspection page.

Meanwhile, the inspection page is where the user will perform fruits freshness inspection (see Figure 4.8). The user can take detect the fruit in real time, then the system will first detect the fruit. The freshness model will run and the result will be shown to the user such as fruit type, fresh/not fresh, once the fruit type is detected the user

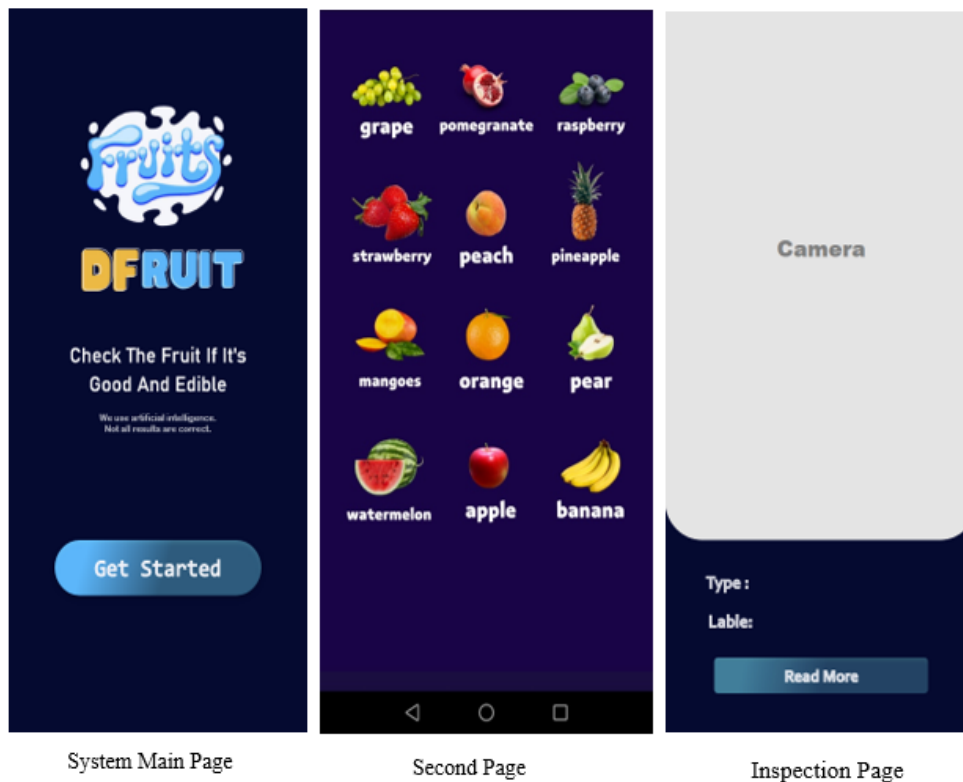


Figure 4.5: Application Main Pages

will get suggestions to read more about the selected fruit. However, if the user has selected an object or not a fruit within the scope of the 12 fruits (apple, banana, mango, orange, watermelon, pear, strawberry, peach, pineapple, blueberry, grape, pomegranate), the selected object/fruit will not be detected. The information page (see Figure 4.9) provides the user with detailed information about the selected fruit, including its nutritional content and the benefits it provides.

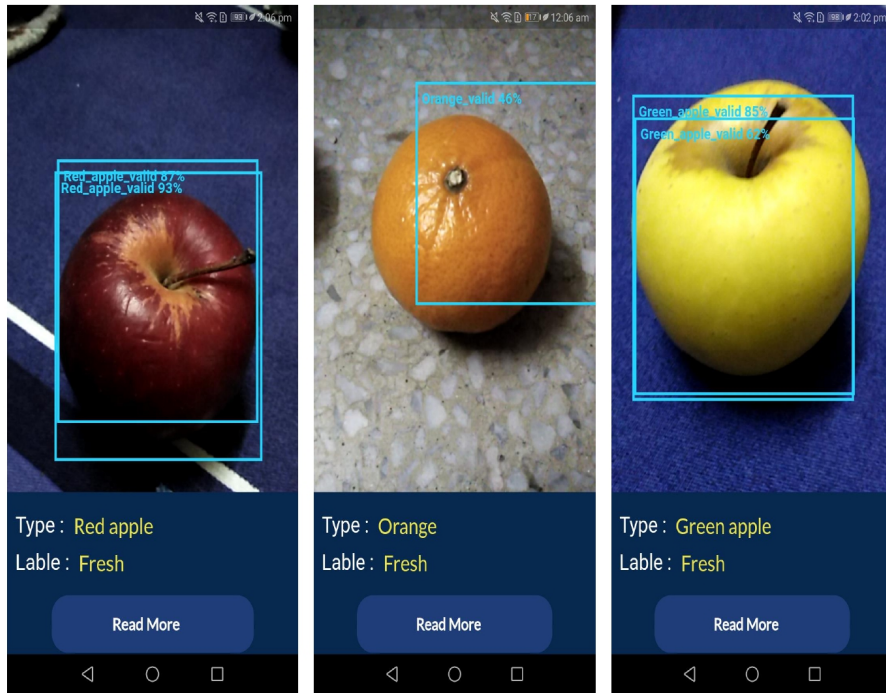


Figure 4.6: Valid Fresh Cases Test

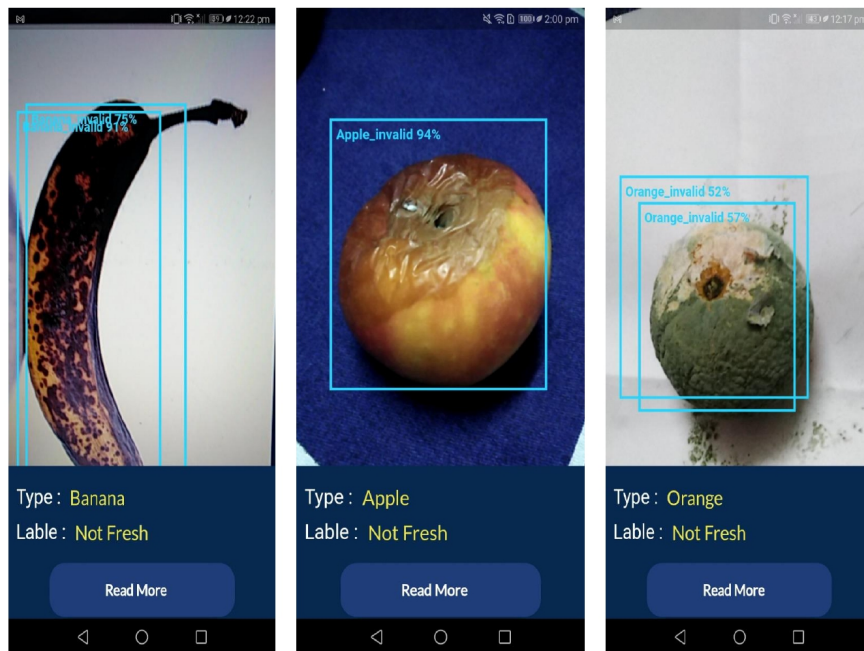


Figure 4.7: Valid Not Fresh Cases Test

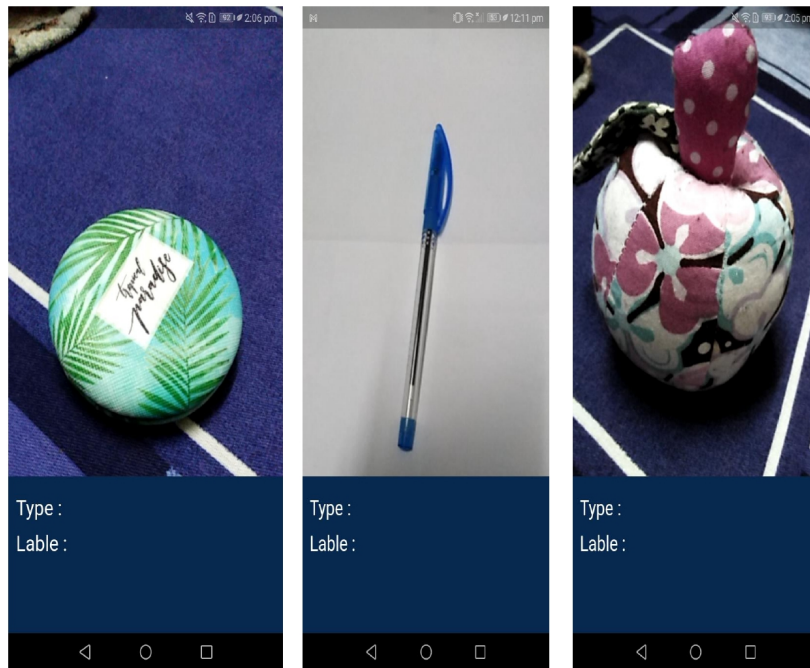


Figure 4.8: Invalid Cases Test

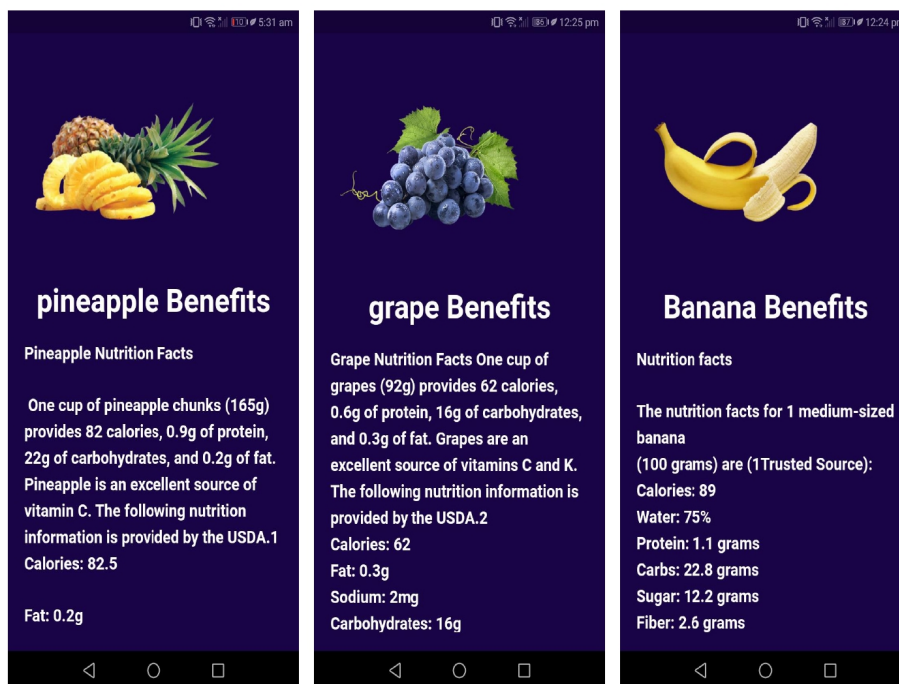


Figure 4.9: Fruit Information Pages

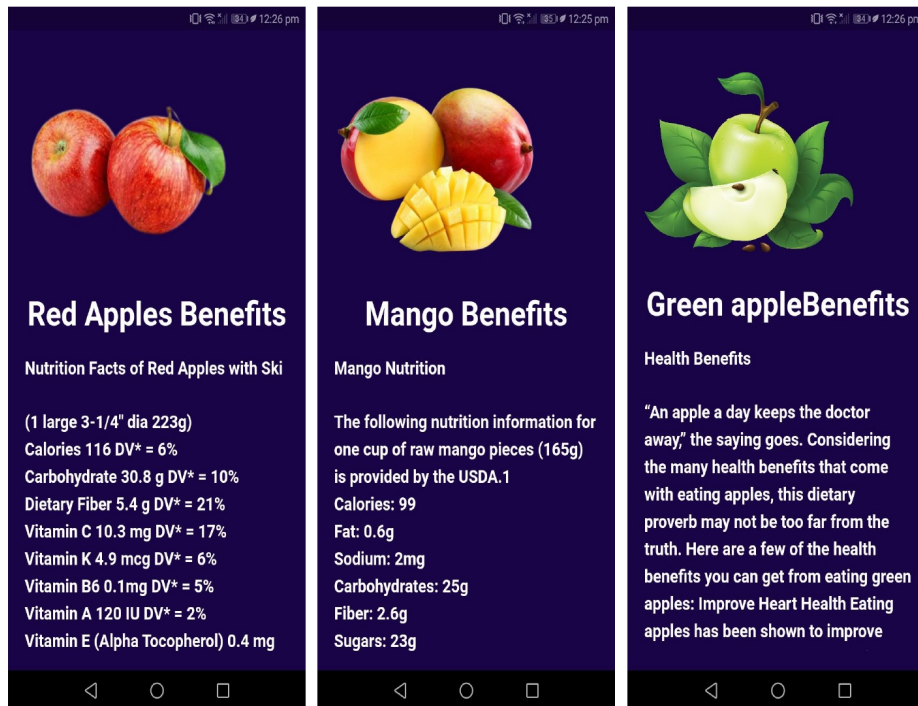


Figure 4.10: Fruit Information Pages (continue)

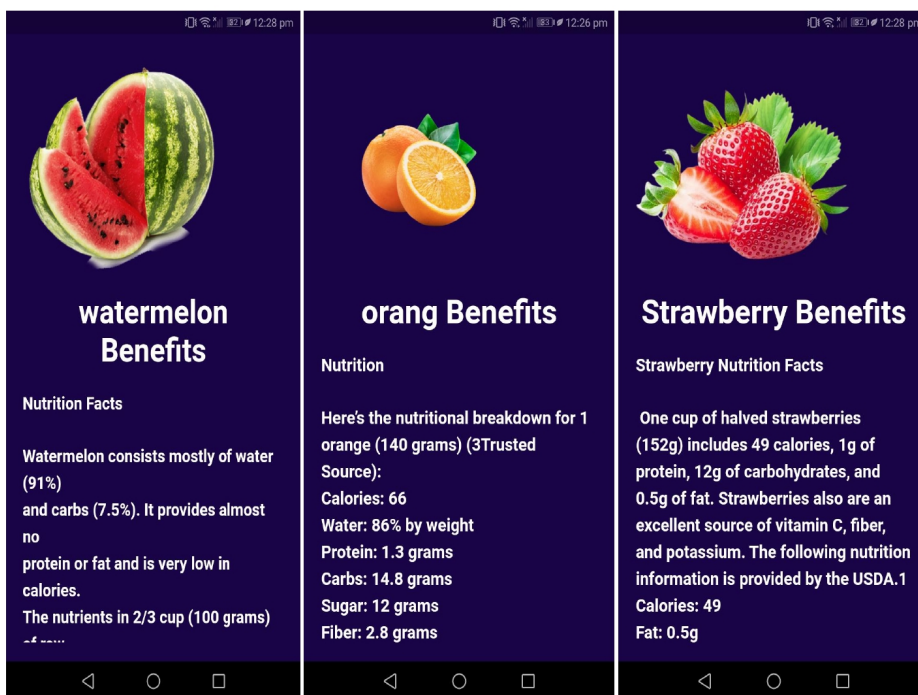


Figure 4.11: Fruit Information Pages (continue)

## 4.4 Evaluation on Real Data

### 4.4.1 Valid Fruit Test Cases

As shown in Figure 4.12, 4.13, and 4.14, 22 tests were performed, 20 were successful while 2 were unsuccessful. One of the failed cases was due to incorrect prediction; the test involved the prediction of not fresh green grapes

as fresh. The chosen fruit was overripe, rather than rotten or not fresh, and the model did not provide the expected result. Overall, the test results show a relatively high level of accuracy, with a success rate of 90.91 percent. Another test case that did not produce the desired results was one in which the fruit classification model incorrectly identified a raspberry as a grape. However, the model has performed well in most of the test cases and can therefore still be used in the final release of the system.

#### **4.4.2 Invalid Fruit Test Cases**

The fruit detection model in the application is performing well, as shown in Figure 4.15 4.16, there were a total of 14 test cases run. 12 of them passed, while 2 of them failed. Two incorrect classifications, the first case classified sliced red guava as watermelon and the seeds in the papaya as black raspberries. This model is used to identify the type of fruit in the input image provided by the user and load the corresponding model if it is available. If the fruit is not recognized, no inspection is performed. Overall, the fruit detection model is functioning properly and will be included in the final version of the system.









<i>Image</i>	<i>Actual Case</i>	<i>Predicted Result</i>	<i>Predicted Type</i>	<i>True/False</i>
	Not Fresh	Not Fresh	Banana	True
	Fresh	Fresh	Banana	True
	Not fresh	Not Fresh	Red Apple	True
	Fresh	Fresh	Red Apple	True
	Not Fresh	Not Fresh	Green Apple	True
	Fresh	Fresh	Green Apple	True
	Not Fresh	Not Fresh	Pomegranate	True
	Fresh	Fresh	Pomegranate	True

Figure 4.12: Valid Test Cases (1)











	Not Fresh	Not Fresh	Watermelon	True
	Fresh	Fresh	Watermelon	True
	Not Fresh	Not Fresh	Mango	True
	Fresh	Fresh	Mango	True
	Not Fresh	Not Fresh	Black Grapes	True
	Fresh	Fresh	Black Grapes	True
	Not Fresh	Fresh	Green Grapes	False
	Fresh	Fresh	Green Grapes	True

Figure 4.13: Valid Test Cases (Continue)



Not Fresh	Not Fresh	Pineapple	True
Fresh	Fresh	Pineapple	True
Not Fresh	Not Fresh	Grapes	False
Fresh	Fresh	Blackberry	True
Not Fresh	Not Fresh	Strawberry	True
Fresh	Fresh	Strawberry	True

Figure 4.14: Valid Test Cases








<i>Image</i>	<i>Predication</i>	<i>True/False</i>
	Watermelon	False
	Black Raspberries	False
	No prediction	True
	No prediction	True
	No prediction	True
	No prediction	True
	No prediction	True

Figure 4.15: Invalid Fruit Cases








	No prediction	True
	No prediction	True
	No prediction	True
	No prediction	True
	No prediction	True
	No prediction	True
	No prediction	True

Figure 4.16: Invalid Fruit Cases

# **CHAPTER 5**

## **TESTING AND EVALUATION**

The purpose of the testing phase is to evaluate the functionality, reliability, and performance of the application. This is accomplished through a series of tests that aim to identify any defects or issues that may exist within the application. Testing is an integral part of the development process, as it helps to ensure that the application meets the requirements and expectations of the users. There are various types of testing performed, including functional testing, performance testing, system testing, and acceptance testing. Each type of testing has a specific focus and is used to verify different aspects of the application. In this chapter, we will describe the testing approach that was taken for the application and provide an overview of the different types of testing that were conducted. We will also discuss the tools and technologies that were used to support the testing process and present the results of the tests. By the end of this chapter, the reader should have a clear understanding of the testing process and the overall quality of the application.

### **5.1 Functional Testing**

Functional testing is a type of testing that verifies that the application under test is functioning correctly and meets the requirements specified in the functional specifications. It involves testing the individual functions of the application to ensure that they are working correctly and producing the expected output. Functional testing is typically focused on verifying the functionality of the application from the user's perspective, meaning that it tests the application's capabilities and features as they are intended to be used by the end user. This type of testing is typically done manually, although some functional tests may be automated using tools such as automated testing frameworks. Functional testing is an important part of the testing process, as it helps to ensure that the application is working correctly and meets the needs of the users. It is typically performed early in the testing process, before other types of testing such as performance or security testing are carried out.

#### **5.1.1 Black Box Testing**

Blackbox testing is a type of testing in which the tester does not have any knowledge of the internal workings of the system under test. The tester treats the system as a "black box" and tests the system based on its input and output, without considering how the input is processed or how the output is generated. Blackbox testing is typically used to validate the functional requirements of the system, as well as to test the system's external interfaces and user

interactions. It is a useful technique for testing complex systems, as it allows the tester to focus on the system's behavior rather than on its internal implementation.

The following categories were tested using blackbox testing techniques in order to identify errors:

- Incorrect Functionality.
- Graphics Defects.
- Variable Errors.
- Performance Issues.

### **5.1.2 White Box Testing**

Whitebox testing is a type of testing in which the tester has knowledge of the internal workings of the system under test and tests the system based on this knowledge. This includes testing the internal logic and structure of the system, as well as the internal data structures and algorithms used. Whitebox testing is typically used to validate the internal code and logic of the system, as well as to test the system's internal interfaces and data structures. It is a useful technique for testing complex systems, as it allows the tester to focus on the system's internal implementation rather than just its behavior.


During white box testing, the following issues were addressed:

- Ensuring that the selected input label matches the classification.
- Verifying that the parameters and arguments are correctly attributed.
- Evaluating the correctness of image attributes.

### **5.1.3 Performance Testing**


Performance testing is a type of software testing that is used to evaluate the performance of a software application or system under a particular workload. It is designed to determine how the software performs in terms of responsiveness, stability, and scalability under different workloads and conditions. Performance testing can be conducted at various stages in the development process, including unit testing, integration testing, and system testing. It is typically focused on testing non-functional requirements such as performance, reliability, and scalability, rather than functional requirements such as correctness or completeness.


To verify the correctness and completeness of the mobile application, a version of the app was released to a small group of users (10 people) 5.1 for testing purposes. This group was asked to test the app and provide feedback, as shown in 5.2. After developing a new version of the app, it was released to the same group of users for further



## App Performance Test

*The purpose of this survey is to gain insight into the actual performance of an application designed to identify the type of fruit in real-time. The survey data collected will assist in identifying any issues or problems that users may be experiencing with the application, which will in turn aid in the identification and correction of errors, as well as the development of a user-friendly final model. Participants are requested to complete the tasks outlined below, and provide feedback on their level of success in doing so.*

 aabc52198@gmail.com (not shared) [Switch account](#)



The App open fast

Yes

No

The second page "Fruit can be tested" is viewed

Yes

No

Figure 5.1: Questionnaire Sample

testing and feedback. The results of the test indicated that some users had difficulty identifying certain fruit, with two out of 10 cases resulting in a slow selection or a wrong classification.

#### 5.1.4 Data Testing

Table 5.1 presents the results of a data testing process that was conducted to evaluate the accuracy of a developed model. This process involved using the model to make predictions based on a set of input data, and then comparing those predictions to the actual outcomes. By comparing the model's predictions to the true outcomes, we were able to determine the extent to which the model was able to accurately capture the underlying patterns in the data. Through this testing process, we were able to gain a better understanding of the strengths and limitations of the model. We were also able to identify any areas where the model could be improved, such as by fine-tuning the

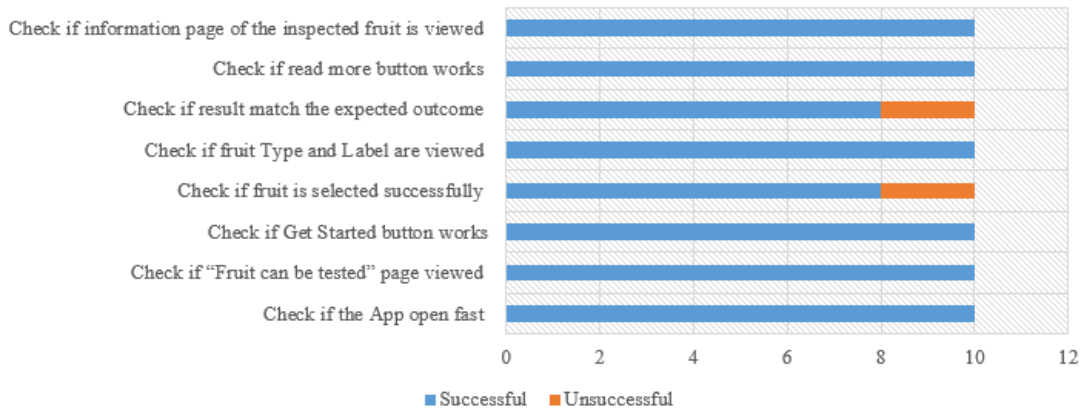


Figure 5.2: Questionnaire Result Sample

model's parameters or by adding additional data to the training set. Overall, the results of the data testing process provide valuable insights into the effectiveness of the model and help us to understand how it can be used to make accurate predictions in the future.



Table 5.1: Data Testing

Class Id	Name	Accuracy	TP	FP
0	Apple_invalid	70.96%	52	56
1	Banana_invalid	58.87%	40	66
2	Banana_valid	66.40%	38	29
3	Bananas_invalid	66.30%	40	3
4	Bananas_valid	59.66%	44	6
5	Black_grapes	55.74%	49	40
6	Black_raspberries	61.35%	51	42
7	Grapes_invalid	76.34%	45	44)
8	Green_apple_valid	71.58%	103	16
9	Green_grapes_valid	69.02%	50	91
10	Mango_invalid	83.09%	65	41
11	Mango_valid	75.93%	158	31
12	Orange_invalid	90.59%	69	90
13	Orange_valid	65.86%	133	25
14	Pear_invalid	75.69%	68	126
15	Pear_valid	77.84%	145	53
16	Pineapple_invalid	94.92%	69	111
17	Pineapple_valid	58.98%	62	18
18	Plum_invalid	86.26%	91	54
19	Plum_valid	74.65%	210	40
20	Pomegranate_invalid	65.71%	62	148
21	Pomegranate_valid	66.37%	127	74
22	Raspberries_invalid	80.89%	52	104
23	Red_apple_valid	70.67%	101	25
24	Red_raspberries	60.91%	50	119
25	Strawberry_invalid	76.56%	106	48
26	Strawberry_valid	58.70%	130	102
27	Watermelon_invalid	81.24%	73	182
28	Watermelon_valid	75.40%	103	42
29	Yellow_apple_valid	77.35%	117	54

# CHAPTER 6

## CONCLUSION AND MAIN OUTCOMES

### 6.1 Project Review

The project has successfully achieved all objectives, including the creation of a Deep Learning model for fruit classification by type and freshness.

### 6.2 Novelties and Contributions

There are several unique aspects in this project which are:

- One aspect of the project involved creating and compiling a dataset of fruit images that were organized by freshness level. This dataset will be useful for future research and development by data scientists, as it will save them time and effort by providing a ready-made dataset rather than requiring them to collect one from scratch.
- The model classifies fruits by their freshness level and incorporated this technology into a mobile application, making it portable and accessible to users. This allows any consumer with the application installed to use it to assess the freshness of fruits during their grocery shopping trips, assisting them in their decision-making process when selecting fruits to purchase.

#### 6.2.1 Contributions

There are currently only a few mobile applications that offer fruit freshness detection, and as previously reviewed, these existing applications have limitations in terms of performance and complexity. The application developed in this project aims to provide a more effective and user-friendly solution for consumers to select fresh fruits before purchasing them, particularly for those who may not have experience in selecting fresh produce. The application detects the freshness of a variety of fruits which other existing applications don't provide. Users simply need to point the phone on the fruit to perform an inspection, and the result of the inspection will be displayed to the user after the selected fruit is processed through the appropriate model.

### **6.3 Limitations**

One of the challenges faced during this project was exporting the Deep Learning model to the Android platform. Various alternatives and configurations were considered for converting the TensorFlow model to a TensorFlow Lite model that could run on Android devices, and found that some trial and error was required to determine the most appropriate conversion method that would maintain the accuracy of the model. Another challenge was collecting a sufficient dataset to develop the Deep Learning model. In order to create a model that would perform accurate fruit inspections in the real world, a large amount of data was necessary.

### **6.4 Future Work**

There are opportunities to improve the fruit freshness detection mobile application by adding more types of fruit to the list of those it can inspect, as well as by enhancing the user interface and design. This can be achieved by collecting additional datasets of the desired fruits, training and retraining the deep learning models, and implementing any necessary updates to the classification process. Additionally, adding a dark mode and further beautifying the design can enhance the user experience.

# Bibliography

- Alzubaidi, Laith et al. (2021). “Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions”. In: *Journal of big Data* 8.1, pp. 1–74.
- Bochkovski, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao (2020). “Yolov4: Optimal speed and accuracy of object detection”. In: *arXiv preprint arXiv:2004.10934*.
- Dhiman, Bhumica, Yogesh Kumar, and Munish Kumar (2022). “Fruit quality evaluation using machine learning techniques: review, motivation and future perspectives”. In: *Multimedia Tools and Applications*, pp. 1–23.
- Gai, Rongli, Na Chen, and Hai Yuan (2021). “A detection algorithm for cherry fruits based on the improved YOLO-v4 model”. In: *Neural Computing and Applications*, pp. 1–12.
- Gunathilake, HRP, CS Perera, and SKWPS Jayasekara (2018). “Fruit detection and classification using machine learning: A review”. In: *Expert Systems with Applications* 94, pp. 280–293.
- Haenlein, Michael and Andreas Kaplan (2019). “A brief history of artificial intelligence: On the past, present, and future of artificial intelligence”. In: *California management review* 61.4, pp. 5–14.
- Islam, Md Rafiqul and Muhammad A Imran (2019). “A Deep Learning Approach for Fruit Freshness Detection”. In: *Sensors* 19.24, p. 5261.
- Karim, Mohammad A, Muhammad A Imran, and Mohammad M Rahman (2019). “Fruit Freshness Detection Using Deep Learning and Transfer Learning”. In: *Sensors* 19.21, p. 4541.
- Kattenborn, Teja et al. (2021). “Review on Convolutional Neural Networks (CNN) in vegetation remote sensing”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 173, pp. 24–49.
- Koirala, Anand et al. (2019). “Deep learning–Method overview and review of use for fruit detection and yield estimation”. In: *Computers and electronics in agriculture* 162, pp. 219–234.
- Lee, Sang-Hyun (2021). “A Study on Fruit Quality Identification Using YOLO V2 Algorithm”. In: *International Journal of Advanced Culture Technology* 9.1, pp. 190–195.
- Mazen, Fatma and Ahmed A Nashat (2019). “Ripeness classification of bananas using an artificial neural network”. In: *Arabian Journal for Science and Engineering* 44.8, pp. 6901–6910.
- Murthy, M S R, S R K Prasad, and V R V Krishna (2019). “Deep Learning for Fruit Freshness Detection Using Convolutional Neural Networks”. In: *arXiv preprint arXiv:1912.05281*.

- Mythri, KJ et al. (2020). "Android Based Ripening Stage Identification for Peppercorns". In: *International Conference on Computer Networks and Inventive Communication Technologies*. Springer, pp. 138–145.
- Pande, Aditi et al. (2019). "An efficient approach to fruit classification and grading using deep convolutional neural network". In: *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*. IEEE, pp. 1–7.
- Roy, Kyamelia, Sheli Sinha Chaudhuri, and Sayan Pramanik (2021). "Deep learning based real-time Industrial framework for rotten and fresh fruit detection using semantic segmentation". In: *Microsystem Technologies* 27.9, pp. 3365–3375.
- Sa, Inkyu et al. (2016). "Deepfruits: A fruit detection system using deep neural networks". In: *sensors* 16.8, p. 1222.
- Salah, AK and MA Hassan (2016). "A review of machine learning techniques for fruit detection and classification". In: *International Journal of Advanced Computer Science and Applications* 7.2, pp. 92–98.
- Zhao, Jiayue and Jianhua Qu (2019). "A detection method for tomato fruit common physiological diseases based on YOLOv2". In: *2019 10th international conference on Information Technology in Medicine and Education (ITME)*. IEEE, pp. 559–563.