FRUIT QUALITY CLASSIFICATION SYSTEM USING IMAGE PROCESSING

BY

OKWOR NMASICHI GRACE
GOU/U22/CSC/885

DEPARTMENT OF COMPUTER SCIENCE AND MATHEMATICS, FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY, GODFREY OKOYE UNIVERSITY, ENUGU STATE.

IN PARTIAL FULFILLMENT OF THE AWARD OF BACHELOR OF SCIENCE (B.SC), DEGREE IN COMPUTER SCIENCE.

SUPERVISOR: ENGR. CAMILUS NJOKU

MAY, 2026.

APPROVAL PAGE

This research, titled **FRUIT QUALITY CLASSIFICATION SYSTEM USING IMAGE PROCESSING** has been assessed and approved by the Department of Computer Science and Mathematics Committees in the Faculty of Computing and Information Technology, Godfrey Okoye University, Enugu, Nigeria.

ENGR. CAMILUS NJOKU	**.....**	**.....**
Supervisor	Signature	Date
External Examiner	**.....**	**.....**
External Examiner	Signature	Date
Dr. Frank Okebanama	**.....**	**.....**
HOD, Department of Computer Science and Mathematics	Signature	Date
Prof. I. A Ajah	**.....**	**.....**
Dean, Faculty of Computing And Information Technology.	Signature	Date

CERTIFICATION

I certify that I completed the research included therein and that it was primarily based on my observations and investigation. Consequently, I will fully accept the University's decision if I am found to have cheated on the assignment.

_____

OKWOR NMASICHI GRACE
GOU/U22/CSC/885

Dedication

This project is dedicated to my eldest brother Zikora Okwor and my family, whose love, support, encouragement and prayers have been instrumental in the successful completion of my academic journey.

ACKNOWLEDGMENTS

First and foremost, I give all glory and honour to God Almighty, whose grace, strength, and wisdom sustained me throughout this academic journey.

I wish to express my sincere gratitude to my supervisor, Mr. Camilus Njoku, for his guidance, patience, and invaluable feedback throughout the course of this project. His expertise and constructive criticism shaped this work into what it is today.

I also extend my appreciation to the Head of Department, Dr. Frank Okebanama, and all the lecturers in the Department of Computer Science and Mathematics, Godfrey Okoye University, for the knowledge and academic foundation they provided over the course of my studies. Their dedication to teaching has equipped me with the skills necessary to undertake and complete this research.

To my parents and brothers, I owe a debt of gratitude that words cannot fully express. Thank you for your unwavering love, sacrifices, financial support, and endless encouragement throughout my academic journey. Every step I have taken has been possible because of the foundation you built for me.

To my friends and colleagues who offered words of encouragement, assistance, and moral support during the most challenging moments of this programme, I am deeply grateful.

Finally, to everyone who contributed in any way, directly or indirectly, to the completion of this project, your kindness and support are sincerely appreciated.

ABSTRACT

Post-harvest fruit losses have continued to be a big agricultural problem for developing nations because manual quality inspection has been found to be unreliable due to its subjective nature, vulnerability to human fatigue, lack of scalability with increased production levels, and failure to evaluate quality condition and ripeness stage together. This paper presents the development of a Fruit Quality Classification System through image processing that could automatically identify and classify fruits in terms of their quality condition and ripeness stage using digital images. A two-model pipeline consisting of a YOLOv8n object detection algorithm and Dual-Head EfficientNet-B0 classifier was created, wherein the former detects and classifies fruit images, while the latter utilizes a multitask learning framework in determining quality (Fresh/Rotten) and ripeness stage (Unripe, Ripe, Overripe) of fruit images. Both models were fine-tuned using transfer learning approach on 13,599 annotated images acquired from Kaggle Fruits Fresh and Rotten for Classification database and divided into 10,901 training and 2,698 test sets using augmentations by Albumentation library and AdamW optimizer with cosine annealing schedule. The evaluation of the dual-head efficient net showed ideal values of the accuracy, precision, recall, and F1-score of 1.0 when assessing quality and ripeness classes. For the YOLOv8n object detector, the precision and recall scores of 1.0, as well as the mean average precision at IOU=50 and at IOU 50-95, scored 0.995 on each of the six fruit classes tested. The entire architecture was implemented as a full-stack web application consisting of a React.js client-side frontend, an Express.js server-side backend, as well as the Flask inference service for the ML algorithms, with a Supabase relational database providing JWT-authentication for the users, persistence in classification history and PDF report generation. Twelve test scenarios for the user interface were completed successfully. The findings show that the implementation of object detection and dual-attribute classification simultaneously in a web application resolves two recurring gaps from the review of literature namely joint quality and ripeness assessment, as well as deployment of a web infrastructure, thereby providing a practical and reproducible contribution to the automation of post-harvest fruit quality management in agricultural and food processing contexts.

Table of Content

List of Tables

List of Figures

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

Agricultural activities are the most significant sources of food and income generation in many countries worldwide, including Nigeria, where agriculture supports the livelihood of many households. The main benefit of agricultural goods lies in the fact that they are highly nutritional, demanded by consumers, and sold on the market at prices which largely influence farmers' profits.

However, one of the most challenging issues faced by both farmers and food vendors is the perishable nature of agricultural products. For instance, fruits are perishable items that can go bad in a short period of time due to lack of checking and sorting. One should note that, The Food and Agriculture Organisation of the United Nations (FAO, 2011) reports that approximately one out of every three food items produced worldwide is lost or wasted and the majority of this waste occurs after the harvesting process due to poor inspection and quality control of fruits (FAO, 2019). In addition, the post-harvest loss of fruits and vegetables in developing countries amounts to up to 30-50 percent of total production (FAO, 2011).

The traditional method of inspecting fruits involves manual checking by farmers and food distributors. The individual picks the fruit, examines the colour of the fruit, analyses the surface of the fruit for any spots, marks or other damage, and determines whether the fruit is fresh or rotten. Even though the process of assessing the freshness of fruits has been carried out for many decades, there are several disadvantages associated with the practice. First of all, the process is time-consuming, requires great effort, and is subjective. The individual who was working many hours may begin making mistakes due to fatigue. Such inconsistency leads to a situation where rotten fruits might end up being consumed by the consumer and vice versa, some good fruits will be discarded (Singh et al., 2022).

With the help of computers and artificial intelligence (AI), new systems have emerged that are capable of analysing images of fruits and deciding whether the produce is fresh or not, without any need for a person to inspect it. Computer vision allows the device to examine an image in the same manner as human eyes can see and understand the meaning of a picture.

Currently, one of the key features in computer vision and artificial intelligence technologies is a deep learning approach. a type of AI that teaches computers to recognise patterns in images by showing them thousands of examples (LeCun et al., 2015).

In previous research works, Convolutional Neural Network (CNN) models have been utilized by many researchers to categorize fruits into fresh and rotten classes with extremely high precision. These models have the ability to recognize visual patterns such as changes in colour, discolouration on surface, bruised areas, and mould, just like human inspectors do, but in a consistent and faster manner. There exists a unique model known as YOLOv8 (You Only Look Once, version 8), which not only detects objects such as fruits, but it can also pinpoint their location within an image while categorizing them.

A Fruit Quality Classification System employing image processing and deep learning approaches is hence developed in this research work. The proposed system uses a YOLOv8n object detection model that locates the fruit images uploaded by users and Dual-Head EfficientNet-B0 model to categorizes each one into a 'Fresh' or 'Rotten' class and determines their ripeness stages. The fruits used in this study include apples, bananas, and oranges, some of the most popular fruits eaten all around the world including in Nigeria.

1.2 Statement of the Problem

Though fruits are an integral part of our daily diet and have economic significance, the process of evaluating their quality is primarily a manual procedure. In general, human inspectors are used to categorize the quality of fruits visually, yet this system has several flaws that could impact both consumers and fruit growers.

The first drawback of this approach is its subjectivity, i.e., two inspectors might come up with two different opinions about the same fruit, making the quality standards inconsistent in the long run. Moreover, the system cannot be scaled easily because the increase in the number of fruits to be processed also means an increase in the number of human inspectors needed for it, hence raising the cost of the whole inspection process significantly. Finally, the system can be affected by inspector's fatigue, which often happens during fruit processing on an industrial scale.

Moreover, most computer-based solutions proposed in scientific literature can only categorize images, which means that all information is taken from one image, regardless of its contents. It might be inefficient since there can be several fruits presented in one picture, each

being in a different state. Also, there is no solution that evaluates both quality and ripeness stage of the fruit at the same time.

Finally, many research systems are not developed to work as a functional application. Instead, these systems are just prototypes that are not ready for implementation in practice by farmers and food processing industries. It would be necessary to develop a fruit quality classification system, which can be fully functional, accurate, and accessed through a user-friendly interface.

1.3 Aim and Objectives of the Study

The aim of this study is to develop a Fruit Quality Classification System Using Image Processing. To achieve the stated aim, the following specific objectives were pursued:

1. Analyse existing manual fruit quality inspection methods and collect fruit image data for classification.
2. Train a hybrid deep learning model for extracting relevant features and fruit classification.
3. Develop a user-friendly web application interface for the proposed system.
4. Evaluate the system using appropriate metrics.

1.4 Significance of the Study

The study is of great importance to various persons and organizations.

For farmers and food industries, the proposed system offers an efficient solution for evaluating the quality of fruits in terms of their freshness and condition without the need for manual inspections. Post-harvest losses can be avoided, thus, guaranteeing that fruits of good quality are delivered to the end-user. Furthermore, it becomes easier to determine when a certain fruit should be marketed or processed.

For researchers and computer science students, the project serves as a case study in the design and deployment of object detection and multi-task deep learning techniques using transfer learning on well-known models like EfficientNet-B0 and YOLOv8.

For consumers have the advantage of knowing that rotten fruits will not be sold. There would be no chance of buying unhealthy fruits that could harm one's health.

For developing countries like Nigeria, where post-harvest losses account for a huge chunk of loss within the farming industry, would see great benefits from deploying fruit quality evaluation systems. Such systems will ensure reduced losses, translating to greater incomes and provision of sufficient foods.

1.5 Scope of the Study

This study focuses on the classification of three types of fruits: apples, bananas, and oranges. These fruits were selected because they are commonly consumed, widely available, and the dataset used for training includes images of both fresh and rotten versions of all three.

The system classifies fruits as either Fresh or Rotten and also estimates the ripeness stage of each detected fruit. The input to the system is a still digital image in JPEG, PNG, BMP, or WEBP format. Real-time video analysis and live camera feeds are beyond the scope of this study.

The dataset used for training and evaluation was obtained from the Kaggle platform and contains 13,599 images of apples, bananas and oranges in both fresh and rotten conditions. The system is tested under controlled photographic conditions similar to those in the dataset. Testing under real-world farm or market conditions is outside the scope of this work but is recommended for future research.

1.6 Limitation of the Study

This study acknowledges the following limitations. First, the training dataset does not include bounding box labels, which means the object detection model was trained using full-image bounding boxes instead of precisely drawn labels around each fruit. This can lead to decreased accuracy in localizing the fruits that appear adjacent to each other in the image.

The second assumption is that the labels for ripeness classes provided in the dataset during training do not exist directly in the dataset itself. Rather, these have been determined indirectly: fresh fruits are labeled 'Ripe', while rotten fruits are labeled 'Overripe'. This implies that the Unripe class will not be well-represented in the data.

Third, the system was tested and trained using images from the controlled laboratory setting having just plain white background. Its performance on the pictures taken in natural environments with complex background, shadows and different light conditions might vary.

Fourth, the model training process was conducted using GPU resources from cloud services (Kaggle) due to the lack of GPU on local machine.

1.7 Operational Definition of Terms

The following terms are used throughout this study and are defined below:

• Artificial Intelligence (AI): The capability of a computer system to execute activities that normally need human intelligence, such as recognising images, understanding speech, and decisions making.

• Deep Learning: A subfield of artificial intelligence wherein computers learn how to do certain tasks based on the analysis of a large amount of data using neural networks.

• Convolutional Neural Network (CNN): A deep learning approach designed specifically for dealing with images. It processes images by analysing their fragments and detecting such features as edges, colours, and objects shapes.

• Image Processing: An application of computer algorithms for the purpose of analysis, enhancement or information extraction from digital images.

• Object Detection: A computer vision technique involving identification of objects contained in an image and drawing a boundary rectangle to outline their location.

• Transfer Learning: A technique implying reusing and adapting an already trained machine learning model on a new dataset and task.

- YOLOv8n: Short for 'You Only Look Once version 8 nano.' It is a fast and lightweight object detection model used in this study to find and identify fruits within uploaded images.
- EfficientNet-B0: A compact and efficient deep learning model used in this study to classify detected fruits as fresh or rotten and estimate their ripeness stage.
- Fresh Fruit: A fruit that is in good condition with no visible signs of spoilage, molding or other damage and can be consumed.
- Rotten Fruit: A fruit that has decayed because of biological, physical, or chemical spoilage and thus cannot be eaten.
- Ripeness Category: The term used to describe how ripe a fruit is. In this experiment, the ripeness of fruits is categorized into Unripe (not yet ripe for consumption), Ripe (ripe

enough for consumption), or Overripe (beyond its best condition). • Bounding Box: A rectangular box around the object in question to indicate its location and dimensions.

- Softmax Confidence Score: A number between 0 and 1 that tells how confident the model is about its prediction. A score of 0.95 means the model is 95% confident.
- ROC-AUC: A measurement of how well a classification model can separate different classes. A score of 1.0 is perfect, while 0.5 means the model is no better than guessing.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter provides an overview of the research conducted by other researchers on issues similar to this research study, such as the use of deep-learning and image processing technologies for detecting quality fruits, the use of object detection techniques in agriculture, and the development of automated food inspection systems. The goal is to understand what has already been done, what works well, what does not and where the gaps are that this study aims to fill.

The chapter is divided into four main sections. Section 2.2 explains the key ideas and technologies that form the foundation of this study. Section 2.3 critically reviews at least nineteen related research works published mostly between 2021 and 2026. Section 2.4 summarises the reviewed works in a table. Section 2.5 identifies the gaps in the existing research that this study addresses.

2.2 Theoretical Background

This section explains the main concepts and technologies that are used in this project. Understanding these ideas is important for appreciating why certain design choices were made in building the Fruit Quality Classification System.

2.2.1 Fruit Quality and Why It Matters

Fruit quality is the state of the fruit as far as its visual state, freshness, smell, taste, and consumable nature are concerned. Fruit quality plays an important role in the market where fruit quality affects the pricing and acceptance of fruit. A fruit that is fresh, has a good **colour** and no apparent signs of damage will fetch more money than a damaged or spoiled fruit. It is for this reason that quality inspection is an important step in the post-harvest processing of fruits.

According to the Food and Agriculture Organisation, approximately one out of every three food produced worldwide is wated or lost (FAO, 2011). Most of this loss occurs after harvest due to poor quality control measures (FAO, 2019). Post-harvest losses in developing nations such as Nigeria can range from 40 percent to 50 percent of the total produce (Fadiji et al., 2023). This leads to income loss to the farmer, shortage of food and food safety problems.

Automating the quality inspection process using artificial intelligence can help reduce these losses significantly.

2.2.2 Conventional (Manual) Methods of Fruit Inspection

Manual inspection has been used for quite some time now in fruit quality inspections. Inspectors inspect the fruits manually by checking and touching them. They also use their senses such as smell to determine whether the fruits are of good quality or not. It relies **heavily** on the expertise of the individual doing the inspections and works well where there are small numbers.

However, the disadvantages of manual inspections include inefficiency, inconsistency and reliance on the alertness of the person. The decision about the same fruit might be different for different inspectors. Workers conducting such inspections get tired due to continuous working thus increasing the errors made. These weaknesses have motivated researchers to develop automated inspection systems that can do the same job faster, more accurately, and more consistently (Singh et al., 2022).

2.2.3 Image Processing and Computer Vision

The process of analysing and understanding digital images is called image processing. In the context of fruit quality inspection, the computer looks at a photograph of a fruit and identifies its characteristics, including colour, texture, form, and visible flaws. These parameters are used to assess the quality of the fruits.

The computer vision technology takes this concept to a whole new level by enabling the computers to visually perceive and interpret their surroundings. The first computer vision algorithms used for inspecting fruits were quite basic and included such steps as colour thresholding (detection of colours in a particular range) and edge detection (identifying borders between objects). Such procedures performed quite adequately in idealized settings yet proved to be inefficient in less controlled conditions (Ukwuoma et al., 2022).

2.2.4 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are another type of neural networks but focused on image processing. Rather than programming the computer for what to look for, a CNN learns how to identify meaningful features without supervision, simply by analysing thousands of images and tagging them. Small filters scan the image and find edges, colors, shapes and textures at multiple levels, identifying the features the image contains.

Nowadays, CNNs are used in agricultural image-related applications as a default model choice. There are various kinds of architectures ranging from old-school approaches, such as VGGNet (Simonyan & Zisserman, 2014), to more modern models like ResNet (He et al., 2016). Moreover, there are lighter or more efficient models like MobileNet (Howard et al., 2017) and EfficientNet (Tan & Le, 2019), which are often applied for agricultural fruit classification, diseases detection and quality control. CNNs are able to distinguish, for instance, a fresh banana from a rotten one through color and texture recognition.

2.2.5 Transfer Learning

The deep learning algorithm must be trained using a huge number of images, and a substantial amount of computational resources are required for training the model from scratch. However, in case of transfer learning, the initial stage is based on the use of an already existing model that was trained on an extensive dataset, typically ImageNet, which comprises more than one million images.

Since the pre-trained model has a knowledge about such basic visual cues as edges and texture detection, training becomes much easier, faster, and consumes much less computational resources. It was proven that the application of transfer learning provides excellent results in case of fruit classification task, reaching up to 90% of accuracy (Kazi & Panda, 2022).

2.2.6 Object Detection with YOLO

Object detection is a computer vision task where the goal is not just to say what is in an image, but also to draw a box (called a bounding box) around each detected object and give its location. This is more difficult than simple image classification because the model needs to find and identify multiple objects in a single image at the same time.

YOLO, which stands for 'You Only Look Once,' is a family of object detection models that is famous for being incredibly fast. In contrast to the earlier two-step detection approach

where regions are initially proposed and then classified, YOLO takes up both the tasks in one go, thereby making it appropriate for use in real-time scenarios. The most recent version of YOLO named YOLOv8 developed by Ultralytics in 2023 is faster and better performing than its earlier versions and has been increasingly used in agriculture applications such as fruit detection (Jocher et al., 2023; Chakrabarti et al., 2024).

2.2.7 EfficientNet and Multi-task Learning

EfficientNet is a CNN architecture developed by researchers at Google. Its uniqueness is based on the fact that it is highly accurate while consuming less computer resources compared to older models such as VGG or ResNet. It happens since EfficientNet scales the depth, width, and resolution of the network in a well-balanced way rather than simply scaling it in one direction (Tan and Le, 2019).

Multitasking learning is the process of training one model to perform several tasks simultaneously. Thus, in this project, the Dual-Head EfficientNet-B0 model is being trained for classification of fruits into fresh or rotten as well as estimation of its ripeness level (unripe, ripe, and overripe). The multitask approach is much more efficient than the separation into several different models and leads to a higher efficiency due to common visual data.

2.3 Review of Related Works

The following section reviews nineteen works related to fruit quality classification and agricultural object detection using artificial intelligence-based food inspection systems.

The research carried out by Kang and Gwak (2022) suggested the development of an ensemble of multi-task deep learning models for the fruit freshness classification. This method was based on the use of transfer learning and the combination of the predictions of a number of CNN models trained on Kaggle fruit dataset. In other words, this work was important in the sense that it proved the effectiveness of multi-task learning for the purpose of classification of freshness. However, their system did not include object detection, meaning it could only classify one fruit per image.

Ukwuoma et al. (2022) provides a comprehensive overview of the applications of deep learning for fruit detection and classification. This study appeared in the journal "Mathematical Problems in Engineering" and included the use of CNNs, YOLO-based methods, and transfer learning applied to fruit quality. From their findings, even though deep learning models yield superior results in laboratory settings, the application of these models in different background

lighting conditions poses a difficulty. This study assisted in identifying the most common models and approaches.

Kumar et al. (2022) suggested an innovative use approach using deep learning techniques for the detecting and classifying fresh and rotten fruits to minimize the loss of food wastage, published in the Journal of Food Quality. They developed a custom CNN model and applied it to different fruit images and their findings were on multi-class classification problems.

The importance of fruit quality inspection using automatic techniques has been stressed in their work that also motivates the current research study. One of the shortcomings was no end-user application development.

Fu et al. (2022) suggested grading techniques for evaluating the freshness of fruits using deep learning-based framework, published in SN Computer Science. They have tested several models of CNN and examined the ability of each model to recognize fruit freshness in different quality grades. It was revealed that colour features are one of the most crucial indicators of fruit freshness, which proves the effectiveness of the image processing technique applied in this paper.

Singh et al. (2022) reviewed the recent advancements in post-harvest loss mitigation using machine learning frameworks, published in the Journal of Food Quality. The review documented how AI and machine learning have been applied globally to reduce post-harvest losses and improve fruit quality management. It provided a wide perspective on the agricultural and economic motivation for this kind of research. The study did not present an original classification system but was important for supporting the background of this project.

Hayat et al. (2024) developed FruitVision, an automated deep learning-based fruit grading system, published in Open Agriculture. The system used CNN models to assign quality grades to fruits based on visual features extracted from images. The study achieved high accuracy on a multi-class grading task and demonstrated that deep learning grading systems can be practically deployed. A key limitation was that the system evaluated whole images without localising individual fruits, making it less useful for images with multiple fruits.

Xiao et al. (2023) published a comprehensive overview in the detection of fruit and recognition through deep learning in agriculture for automation in harvesting in the journal *Agronomy*. In their work, the researchers have reviewed the progress in object detection architectures, mainly concentrating on region-based CNNs and current generation YOLOs. The researchers stated that one of the major challenges in the operation of vision systems in agriculture is the effect

of environmental factors, including occlusions, dense overlapping and natural lighting variations. While the scope of their work focused on automated harvesting rather than quality inspection, their findings indicate that one-stage detectors like YOLOv8 are at the forefront in achieving a compromise between speed and precision. Consequently, this directly validates the selection of the lightweight YOLOv8n architecture in the current study to overcome similar ambient lighting and occlusion challenges during sorting.

Chakrabarti and Sharma (2024) compared the performance of YOLOv3, YOLOv4, and YOLOv8 for fruit detection and classification across multiple fruit categories, published in the International Journal on Recent and Innovation Trends in Computing and Communication. The study found that YOLOv4 performed best in their specific comparison but YOLOv8 also delivered competitive results, particularly in terms of detection speed. The experiment confirmed the efficiency of YOLOv8 for fruit detection tasks and justified its use for this project.

Stasenko et al. (2023) have suggested employing deep learning and near-infrared artificially generated images for the prediction of post-harvest decay of apples, which was described in Entropy journal. Stasenko et al. employed an innovative approach of image generation and image segmentation techniques for the detection of decay on an earlier stage than when it becomes visible on the surface of fruit. Although this approach was more complex and required special imaging devices, it proved that AI could help detect problems with the quality of fruits even before humans did it, which supports the overall vision of this study.

Fadiji et al. (2023) presented a paper in Frontiers in Sustainable Food Systems, in which they outlined the research agenda for the application of artificial intelligence to post-harvest agriculture. They examined how AI, including such approaches as deep learning and computer vision, was used to minimize post-harvest losses around the world. The authors stated that fruit quality assessment based on AI was one of the most promising fields and more studies should be done to make AI practical.

Naranjo-Torres et al. (2023) developed the CNNs model based on actual and artificial image data sets for bananas classification based on ripeness level presented at the VISIGRAPP conference. The findings showed that introducing artificial data to training increased the classification accuracy, particularly when the amount of actual data was small. This is relevant since it shows how to overcome the problem of limited training data for certain ripeness levels such as Unripe class which is underrepresented in the dataset used in this study.

Nerella et al. (2023) compared several deep learning architectures used for the classification of fresh/rotten fruits presented at the RAEEUCCI International Conference. They compared such models as ResNet, VGG, and MobileNet on the common Kaggle fruits dataset by evaluating their accuracy, efficiency, and computational cost. It was found out that lightweight models such as MobileNet demonstrated similar levels of accuracy as heavy ones but had higher efficiency and were better for use. The comparison helped to choose EfficientNet-B0 in this study.

Knott et al. (2023) conducted a research study on machine learning approaches for image-based fruit quality assessment and introduced the idea of 'facilitated machine learning' in order to make deep learning more convenient for non-expert users from the food industry. This paper demonstrated the existence of deployment barriers related to the difficult implementation process, making the development of an easy-to-use interface all the more important for this project.

Aherwadi et al. (2022) conducted a review of fruit quality assessment based on image processing, machine learning, and deep learning approaches in the journal Advanced Applied Mathematics and Sciences. The authors reviewed the advantages and disadvantages of both classical image processing and modern deep learning methods when used for fruit quality evaluation purposes. It was found that deep learning significantly outperforms other methods, while future systems should be able to perform detection and classification at once – exactly what this study suggests.

Dhiman et al. (2022) reviewed fruit quality evaluation using machine learning approaches in the journal Multimedia Tools and Applications. They surveyed more than fifty studies and found that although the machine learning models have high accuracy when tested under laboratory conditions, their accuracy is low in real-life situations owing to variations in the environment. They observed the need for better and more generalized models, stating that the multi-stage pipelines consisting of both detection and classification are more useful in practice. This observation justifies the approach used in this study.

Zhu et al. (2021) published a survey on deep learning models and machine vision applications in food production systems in the journal Current Research in Food Science. The survey covered applications of deep learning in fruit and vegetable quality assessment, grading, defect detection, and freshness estimation. It highlighted the growing role of computer vision in automating food safety and quality control processes. The survey confirmed that CNNs are the

most used and most effective approach for image-based food quality tasks, validating the method used in this study.

Foong et al. (2021) proposed a rotten fruit detection system using ResNet50, presented at the IEEE Control and System Graduate Research Colloquium. The study trained ResNet50 on images of rotten and fresh fruits and achieved high classification accuracy. This work demonstrated that pre-trained CNN backbones are highly effective for binary fresh/rotten classification, reinforcing the use of pre-trained architectures in the current study. A limitation was that the system only distinguished fresh from rotten without spatial detection or ripeness estimation.

Mesa et al. (2021) proposed a multi-input deep learning model combining RGB images and hyperspectral imaging for banana grading, published in the journal Agriculture. The model used two separate input streams to capture both visual and spectral information about the banana. While the use of hyperspectral imaging is beyond the scope of this study, their work demonstrated that colour (RGB) information alone can achieve very good grading accuracy for bananas, supporting the colour-based approach used in this project.

Yuan et al. (2024) proposed an innovative approach for detecting fruit freshness using a deep learning model trained on colour features combined with texture analysis. Their study was published in a peer-reviewed journal and reported accuracy exceeding 94% across multiple fruit types including apples and oranges. The study confirmed that colour remains one of the strongest visual indicators of fruit freshness and that combining it with texture features improves overall detection accuracy, a finding that is consistent with the feature extraction approach used in the Dual-Head EfficientNet-B0 model in this project.

Table 2.1 summarises the key details of the reviewed works, including their contributions, methods, and limitations:

Table 2.1: SUMMARY OF RELATED WORKS

S/N	Author(s) & Year	Contribution / Work Done	Technique / Methodology	Limitations

1	Kang and Gwak (2022)	Ensemble multi-task CNN for fruit freshness classification using Kaggle dataset.	Transfer learning; ensemble multi-task CNN	No object detection; single-fruit images only
2	Ukwuoma et al. (2022)	Comprehensive review of deep learning technique for the detection and classification of fruit.	Review of CNN, YOLO, transfer learning approaches	Review only; no original system built
3	Kumar et al. (2022)	Deep learning frameworks to identify and categorize fresh and damaged fruits to minimize food wastage.	Custom CNN; multi-class fruit images	No deployment interface; no detection
4	Fu, Nguyen and Yan (2022)	Deep learning grading methods for fruit freshness across multiple quality levels.	CNN classification; colour features	No detection or ripeness estimation
5	Singh et al. (2022)	Review of AI and ML for post-harvest loss mitigation and quality management.	Literature review; ML frameworks	No original classification system
6	Hayat et al. (2024)	FruitVision: deep learning-based automatic fruit grading system.	CNN grading; transfer learning	Whole-image classification; no per-fruit detection
7	Xiao et al. (2023)	A comprehensive review of deep learning frameworks for fruit detection and automatic harvesting.	Review of YOLO and R-CNN approaches	Review only; lacks direct focus on post-harvest quality assessment.

8	Chakrabarti et al. (2024)	Compared YOLOv3, YOLOv4, YOLOv8 for multi-fruit detection and classification.	YOLO variants; comparative study	No quality (fresh/rotten) classification included
9	Stasenko et al. (2023)	Deep learning for early post-harvest apple decay prediction using generated images.	CNN segmentation; synthetic image generation	Requires specialised near-infrared hardware
10	Fadiji et al. (2023)	Mapped AI research agenda for post-harvest agriculture applications.	Review; AI and ML applications survey	No original system; agenda-setting paper only
11	Naranjo-Torres et al. (2023)	CNN with real and synthetic images for banana ripeness classification.	CNN; synthetic data augmentation	Single fruit type; no quality classification
12	Nerella et al. (2023)	Compared CNN architectures for fresh/rotten classification on Kaggle dataset.	ResNet, VGG, MobileNet comparison	No detection; no ripeness estimation
13	Knott et al. (2023)	Facilitated machine learning for image-based fruit quality assessment.	ML models; usability focus	No detection; no deployment infrastructure
14	Aherwadi et al. (2022)	Review of image processing, ML, and deep learning for fruit quality identification.	Systematic review	No original classification system developed
15	Dhiman et al. (2022)	Review of fruit quality evaluation using ML in over fifty studies.	Survey of ML techniques for quality	No original system; performance gap in real world

16	Zhu et al. (2021)	Survey of deep learning and machine vision for food processing applications.	Deep learning survey; food processing	Survey only; no fruit-specific system built
17	Foong, Meng and Tze (2021)	Rotten fruit detection using ResNet50 with transfer learning.	ResNet50 transfer learning; binary classification	No spatial detection; no ripeness estimation
18	Mesa and Chiang (2021)	Multi-input deep learning with RGB and hyperspectral imaging for banana grading.	Multi-input CNN; hyperspectral imaging	Requires specialised imaging hardware
19	Yuan et al. (2024)	Innovative deep learning approach for detecting fruit freshness using colour and texture.	Deep learning; colour and texture features	No object detection or ripeness stage estimation

2.4 Research Gap

After reviewing the above studies, the following gaps were identified in the existing literature:

First, most existing studies classify whole images instead of locating and analysing each fruit individually within an image. This is a major limitation because in real-world scenarios, a photograph may contain many fruits of different quality levels. There have been a few studies utilizing object detection as the methodology to tackle this problem, and even they (such as Chakrabarti et al., 2024) have not connected it to the classification of quality or ripeness stage of a fruit.

The second issue with the studies from the review is the fact that none of them deal with fruit classification into fresh/rotten categories and estimating the ripeness of fruits within

the same system. It would seem practical, since knowing whether a particular fruit is fresh and unripe or fresh and ripe or overripe will require both parameters.

Thirdly, most reviewed works present experimental models which lack a complete software product and such aspects as user authentication, history keeping and reports generation, and web interface.

In this research, all these problems are solved through a creation of a comprehensive Fruit Quality Classification System, which consists of (1) YOLOv8n model for fruit detection and localization within an image; (2) Dual-Head EfficientNet-B0 model for simultaneous quality classification and ripeness estimation; and (3) complete web application with all necessary features.

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

3.1 Introduction

This chapter presents the systematic analysis and design of the Fruit Quality Classification System (FQCS). The chapter is organised to reflect the principal phases of structured system development: analysis of the existing system and the problem domain, specification of functional and non-functional requirements, articulation of the design methodology, construction of Unified Modelling Language (UML) behavioural and structural models, and presentation of detailed system design components including input and output specifications, database schema, and system architecture. Together, these sections provide the conceptual blueprint from which the implementation described in Chapter Four is derived.

3.2 System Analysis

3.2.1 Analysis of the Existing System

The current system of quality evaluation of fruits in most agriculture and food processing setups is manual inspection. In this system, trained human inspectors manually examine the fruits or batches and base their evaluation of quality of the produce on various visual aspects including color, blemishes, bruises, mold, physical appearance, and the overall appearance of the fruit.

In smaller scale retail and farming setups, the process is carried out through informal visual examination by either the individual vendor or farmer while in larger scale food processing plants the system follows formalized grading lines involving many human inspectors examining the fruits in turn on conveyor lines. The process is therefore human based and depends on the human visual pattern recognition skills of the trained inspectors who are pressed for time. There is no system of record keeping of quality information which can be used for trend analysis.

3.2.2 Limitations of the Existing System

These limitations in the manual inspection process have been listed as the main reasons behind the suggested system:

i. Subjectivity and Inconsistency: The assessment of quality is inherently subjective, varying from person to person due to his experience, training and perception.

differences. This inconsistency reduces the reliability and comparability of quality assessments, particularly across different inspection sessions or personnel.

ii. Throughput Constraints: Human inspection throughput is limited by attentional capacity and physical processing speed. High-volume post-harvest operations frequently require trade-offs between inspection thoroughness and processing speed, resulting in either bottlenecks or increased error rates.

iii. Fatigue and Attention Degradation: Inspection is a highly cognitive activity that can lead to fatigue. Attention lapses caused by it result in the appearance of a considerable number of errors in the inspection process, which may be hazardous to food security and costly for a company.

iv. Absence of Persistent Records: Manual inspection provides no possibility to create any permanent record of the results that could be used for historical analysis or quality control.

v. No Ripeness Differentiation: Manual inspection is principally focused on defect detection (fresh versus rotten) and does not systematically assess ripeness stage, despite its commercial relevance to optimal timing of retail presentation. Fig. 3.1 presents the flowchart of the existing manual fruit quality inspection process, illustrating the sequential steps followed by human inspectors from post-harvest collection to final sorting

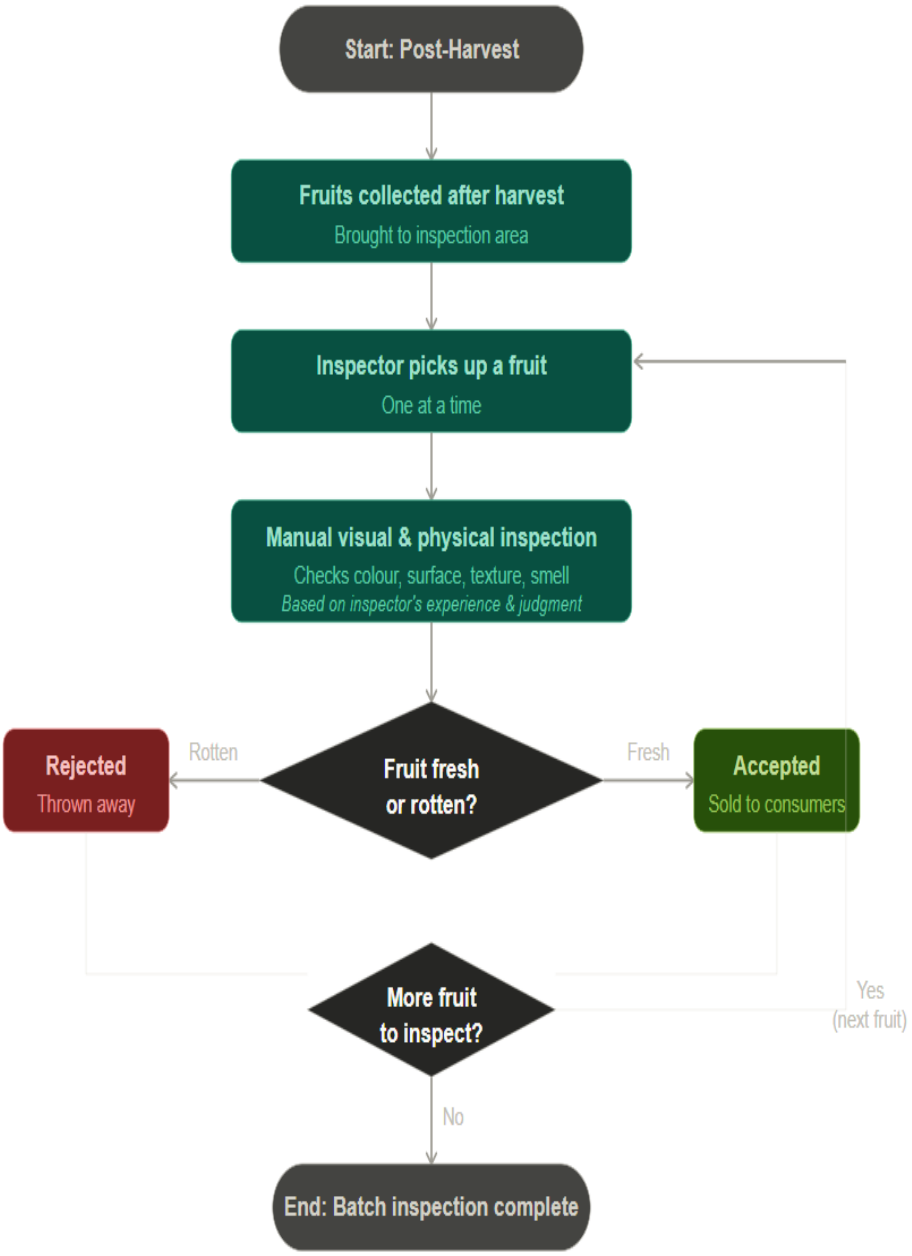

Fig. 3.1: Flowchart of the Existing Manual Fruit Quality Inspection Process

3.2.3 Analysis of the Proposed System

The proposed Fruit Quality Classification System has the following functionalities that overcome the identified limitations: automated fruit regions detection based on image recognition using YOLOv8n model; objective classification of each region as either Fresh or Rotten and estimating the stage of ripeness as Unripe, Ripe or Overripe using fine-tuned Dual-Head EfficientNet-B0 model; storing of all results of the classification process in the

relational database with isolation of information per user enabling historical retrieval and trend analysis; user authentication to protect the confidentiality of the data; PDF reports about the results of the classification process.

3.2.4 Functional Requirements

The system shall satisfy the following functional requirements:

1. User Registration and Authentication: The system shall allow new users to create accounts with their email and password.
2. Upload Image and Validate: The system should allow only authenticated users to upload images of fruits in JPEG, PNG, BMP, or WEBP formats with a maximum file size of 20 MB. This will involve validation of MIME type and file size, along with appropriate error messages for any faulty input.
3. Detection of Fruit and Localisation: This system is expected to use a YOLOv8n model to detect all fruit regions in the uploaded image, and output normalised coordinates of the bounding boxes for the regions.
4. Quality Classification: The system should classify fruit quality into either Fresh or Rotten categories, providing predicted class label and confidence score.
5. Ripeness Prediction: The system is expected to predict the stage of ripeness, i.e., Unripe, Ripe, or Overripe, along with predicted class and confidence score.
6. Result Display: The results of the system should be displayed in terms of three different ripeness labels, i.e., Ripe, Unripe, and Overripe, and two different quality classes: fresh and rotten.
7. Generate Report: The authenticated user should be allowed to download a classification report in the form of a structured PDF file
8. Classification History: The system will store all historical classifications and provide the authenticated user with an ability to browse historical classifications via pagination when picking specific records.
9. History Retrieval: The user will have the ability to view the entire history of any classification that was made in the past, along with all identified regions and their respective predictions.

3.2.5 Non-Functional Requirements

The system shall additionally satisfy the following non-functional requirements:

1. Performance: The whole pipeline of inference (detection, quality classification, and ripeness estimation) shall finish in 30 seconds using CPU hardware without GPU support for a standard image.
2. Security: All API endpoints that require user data will be secured using JWT authentication. Database will use Row Level Security policy that prevents accessing another user's data. Uploaded images will be saved in private Supabase Storage buckets and can be accessed only by the owner.
3. Usability: User interface will be responsive to both desktop and tablet screens, and will notify the user about the progress of async requests.
4. Reliability: The system will handle incorrect data gracefully and respond with structured JSON errors with proper HTTP status code.
5. Maintainability: The codebase will follow the Object-Oriented Design approach where classes have well-defined responsibilities, modules have clear boundaries, and unit & integration testing is done with at least 80% code coverage on key paths.
6. Scalability: The four-tier architecture shall support horizontal scaling of the ML service and backend tiers independently, should demand increase beyond the capacity of a single-instance deployment.

3.3 System Design Methodology

3.3.1 Object-Oriented Analysis and Design (OOAD)

Object Oriented Analysis and Design (OOAD) was identified as the controlling design paradigm for the present study. OOAD focuses on organizing system functionality using objects that are self-contained units consisting of both data also referred to as attributes and behaviour (methods). Inheritance, Polymorphism, and Composition are some of the design concepts used in designing complex software systems.

Reasons for selecting this approach are varied. To start with, the system under development includes several entities such as users, request for classification, detected region, report, and machine learning models that have defined attributes and behaviours which can be modelled as objects. In addition to this, the ML service layer requires the application of an inheritance-based design approach whereby the Base-Model class provides the common interface (i.e., load, pre-process, infer, and postprocess methods) for all model wrappers and the specific implementation of these behaviours is realized in classes such as Fruit-Detector,

QualityClassifier, and Ripeness-Estimator. This design demonstrates Template Method pattern while also satisfying Liskov Substitution Principle. Additionally, the Strategy pattern can be used to apply a PDF report generation strategy.

Furthermore, OOAD produces models such as class diagrams, sequence diagrams, and use case diagrams that naturally communicate system structure to both technical implementers and academic reviewers.

3.4 Dataset Collection, Description and Preprocessing

3.4.1 Dataset Collection

The primary dataset employed in this study, Fruits Fresh and Rotten for Classification, was obtained from the Kaggle open-data platform. The dataset contains 13,599 RGB images pre-partitioned into training (10,901 images) and test (2,698 images) splits, organised across six class directories corresponding to fresh and rotten instances of apples, bananas, and oranges. Images were captured under controlled laboratory conditions at varying orientations and scales.

3.4.2 Dataset Description

The main dataset used in the training and evaluation process consists of the "Fruits Fresh and Rotten for Classification" dataset hosted on Kaggle by Sriram Reddy. The dataset consists of 13,599 high-quality images of both fresh and rotten fruits set against plain white backgrounds, pre-divided into training and test subsets. Table 3.1 contains the basic dataset information used in this study.

3.4.3 Dataset Properties

Table 3.1: Basic Dataset Information

Property	Value
Dataset Name	Fruits Fresh and Rotten for Classification
Source	Kaggle: kaggle.com/datasets/sriramr/fruits-fresh-and-rotten-for-classification
Total Images	13,599
Number of Classes	6 (Fresh Apple, Fresh Banana, Fresh Orange, Rotten Apple, Rotten Banana, Rotten Orange)

Property	Value
Training Images	10,901
Test Images	2,698 (approximately 80/20 split)
Image Format	JPEG
Background	White background (controlled studio photography)
Fruit Types	Apple, Banana, Orange
Quality Labels	Fresh, Rotten (derived from folder name)
Ripeness Labels	Inferred: Fresh = Ripe, Rotten = Overripe
License	Public domain / open use for research

3.4.4 Data Cleaning and Pre-processing

Since images obtained from the real world often have discrepancies like noise, difference in image sizes, illumination changes, class imbalance, corrupted files, and unbalanced colour distribution, which if not properly addressed before training the model, will result in ineffective generalization causing low predictive performance and instability in optimization process.

In order to make sure that the system works efficiently, the following pre-processing procedures were developed to improve consistency in images, increase the learning ability of the model, avoid overfitting, and ensure compatibility with EfficientNet-B0 classification model and YOLOv8n object detection model.

The pre-processing procedure included several steps like dataset verification, class inspection, image normalization, label mapping, resizing, augmentation, tensorizing, statistics, and DataLoader improvement. The stages include:

1. Dataset Exploration and Class Distribution Analysis: Exploratory data analysis was performed to understand the distribution of test and train samples across classes. A detailed class distribution analysis was equally conducted to determine whether the dataset suffered from severe imbalance problems. The analysis revealed that the rotten fruit categories contained slightly more images than the fresh fruit categories. The class distribution table below shows that although some imbalance existed, it was not severe

enough to prevent successful model learning. Table 3.2 presents the class distribution across training and test splits.

Table 3.2: Class Distribution

Class	Train Images	Test Images	Total	Train %	Test %
Fresh Apples	1,693	**419**	2,112	80.2%	19.8%
Fresh Banana	1,581	**391**	1,972	80.2%	19.8%
Fresh Oranges	1,466	**363**	1,829	80.2%	19.8%
Rotten Apples	2,342	**580**	2,922	80.2%	19.8%
Rotten Banana	2,224	**550**	2,774	80.2%	19.8%
Rotten Oranges	1,595	**395**	1,990	80.2%	19.8%
Total	**10,901**	**2,698**	**13,599**	**80.2%**	**19.8%**

2. Label Cleaning and Semantic Mapping: Semantic labelling was introduced because raw folders in the original dataset labels had folder-based categorical names such as freshapples, rottenbanana, freshoranges, etc., and were not suitable. Each original label was decomposed into two separate semantic tasks of Quality Classification and Ripeness Classification allowing the model to simultaneously learn multiple prediction objectives from the same image. For example, freshapples was given the quality label of “Fresh” and the ripeness label of “Ripe” while rottenbanana was given the quality label of “Rotten” and the Ripeness label of “Overripe.” Because neural networks cannot directly process textual labels, the labels were further encoded labeling fresh as 0 and rotten as 1 and for the ripeness task, unripe class was labeled as 0, the ripe class was labeled as 1 and overripe was labeled as 2.

3. Image Resizing and Dimensional Standardization: Images collected from different sources often have varying resolutions and aspect ratios. Therefore, all images were resized to

a standardized resolution according to the model being trained. For EfficientNet-B0 mode, images were resized to 224 × 224 pixels and for YOLOv8n object detection training, images were resized to 640 × 640 pixels.

4. Pixel Value Normalization: Pixel normalization was performed to normalize image pixel values which range from 0 to 255 into smaller numerical ranges that the models can process. This was achieved by converting image intensities into floating-point tensors and scaling them appropriately. In addition to normalization, dataset-specific pixel statistics were computed using approximately 400 sample images. The mean and standard deviations for RGB channels were calculated and compared with ImageNet normalizations due to the fact that the pre-trained model, EfficientNet-B0, was trained using ImageNet. It was found that the dataset contained fruits with brighter and warmer colour distributions than ImageNet since the images of fruits are known to have dominant red, yellow, and orange colours. Normalization helped in optimizing the parameters by keeping them numerically stable throughout backpropagation. Table 3.3 presents the computed RGB channel mean and standard deviation values and compared against ImageNet statistics.

Table 3.3: Pixel Value Normalization

Channel	**Dataset Mean**	**Dataset Standard Deviation**	**ImageNet Mean**	**ImageNet Standard Deviation**
Red	0.7274	0.3302	0.485	0.229
Green	0.6291	0.3416	0.456	0.224
Blue	0.5227	0.3662	0.406	0.225

5. Tensor Conversion: Images were converted to tensors since deep learning models use tensors and not images. The entire process included loading of images, RGB conversion, resizing, augmentation, tensor conversion and normalization.

6. Data Augmentation: This process improves the model's ability to generalize under varying environmental conditions such as different lighting conditions, camera angles, occlusions, scale variations, background differences. This process is extremely important

because deep learning models often overfit when trained on limited datasets. The augmentation techniques implemented included horizontal flipping, random rotation, brightness adjustment, contrast adjustment, scaling transformations, random cropping, and color perturbation. For the YOLOv8n model training, additional augmentations included mosaic augmentation, MixUp augmentation, HSV transformations, Random erasing, and copy-paste augmentation.

7. Batch Preparation and DataLoader Optimization: After preprocessing, the dataset was wrapped inside PyTorch DataLoaders. The DataLoader mechanism handled batch generation, data shuffling, parallel loading, and memory optimization. The configured batch size was 32 images per batch, therefore, the DataLoader produced: Table 3.4 presents the number of batches generated per dataset split.

Table 3.4: Dataset Train and Test Batches

Split	**Batches**
Training Batches	341
Testing Batches	85

Shuffling was enabled during training to prevent the neural network from learning sample ordering patterns. Parallel workers were also utilized to speed up the process of image loading while GPU training.

Specific Preprocessing for YOLOv8n Object Detection: YOLO works on the basis of bounding boxes rather than class labels alone; thus, a preprocessing step is required in order to convert dataset labels into YOLO annotation format to ensure compatabilty with the YOLOv8n training engine.

3.5 System Modelling Using UML

3.5.1 Use Case Diagram

The use case diagram shows one key actor within the system that is the authenticated user. Authenticated User interacts with the system via six main use cases which are Register Account / Login, Upload Image, Classify Quality, View Classification Results, Download Report and View Classification History.

Actors in the System

User: The authenticated user is the main and only actor that interacts with the system. This is any individual who needs to analyse the quality of fruits. The user uploads the fruit image, initiates the process of classification and views the results among other actions.

Use Cases Explained

1. Register/Login: This allows users to create an account and be authenticated into the system
2. Upload Image: This allows users to upload fruit images in various formats such as JPEG, PNG, BMP and WEBP formats
3. Classify Quality: This identifies the quality level such as fresh/rotten using machine learning model
4. View Results: This displays the output of the prediction including quality, ripeness and the confidence score of the prediction.
5. Download Report: This generates downloadable PDF reports
6. View History: This allows users to retrieve previously processed results

Associations

- The user is directly associated with all use cases, meaning that since the user uses every function and there are no other actors present in this system.

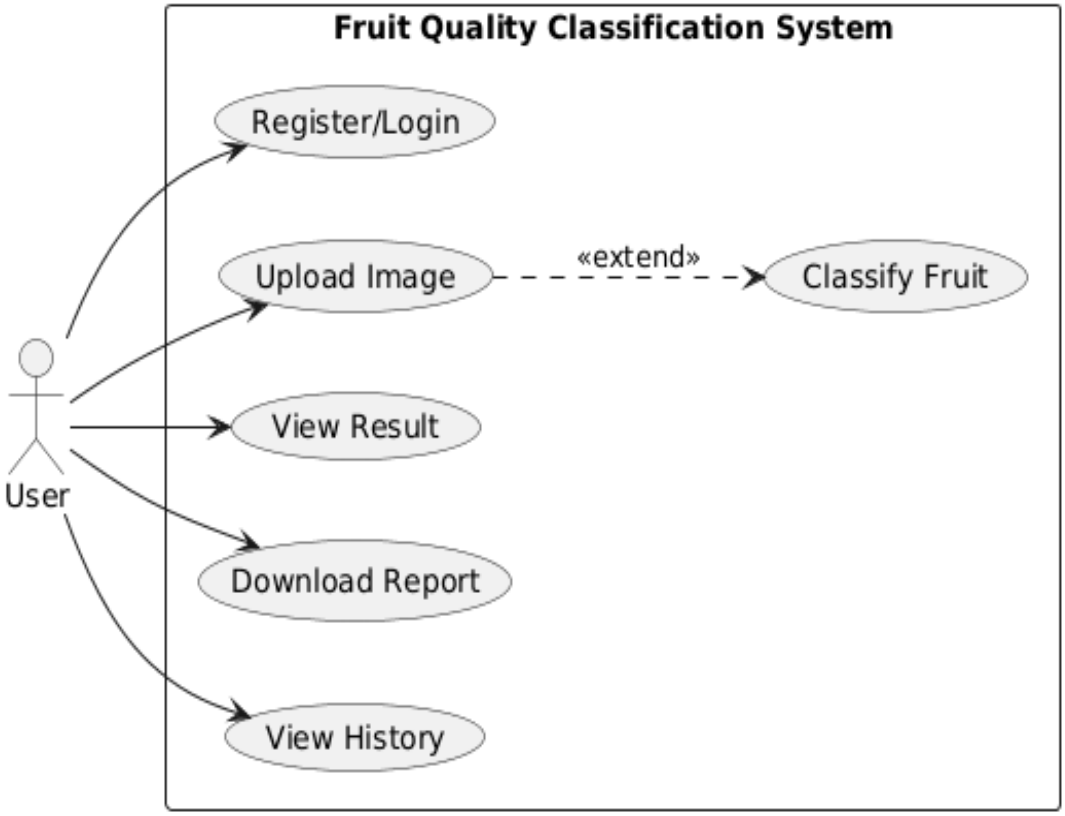

Fig. 3.2: Use Case Diagram

3.5.2 Activity Diagram

The first process of interest is the image classification pipeline. Upon the launch of the process triggered by an authorized user and the upload of the image, the backend verifies the file type and size. Upon successful verification, the image is sent to the ML service to run the detection process, followed by the quality classification process for every detected region, and then the ripeness estimation process (EfficientNet-B0) for every detected region. The results are collected, saved to the database, and shown to the frontend. Additionally, the image is saved to Supabase Storage. the image classification workflow. A failure at any stage results in a structured error response returned to the user.

Flow Explanation

1. Start: Represents the beginning of the process
2. Upload Image: The user uploads an image which is either in JPEG, PNG, BMP or WEBP format.
3. Validate Image: The system checks for file format, and file size
4. Decision Node (Valid?): If the input is valid, the algorithm continues or an error message is displayed.

5. Pre-process Image: This step includes resizing and normalization of the image.
6. Detect Fruit Region: Identify bounding box around fruit
7. Crop Fruit: Extract detected region for classification
8. Classify Quality: Predict quality category
9. Estimate Ripeness: Predict ripeness stage
10. Store Results: Save output in database
11. Display Results: Show results to user
12. End: Process terminates

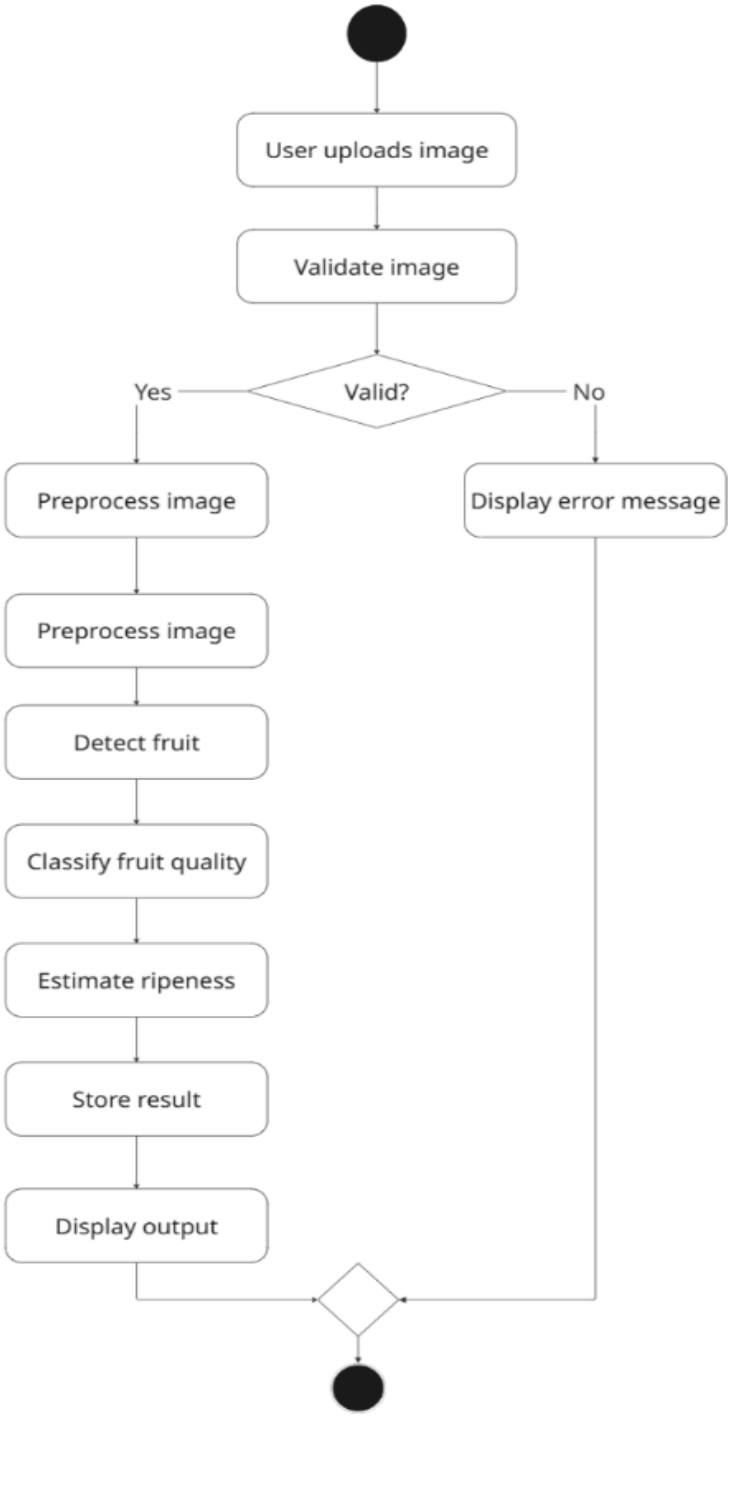

Fig. 3.3: Activity Diagram

3.5.3 Class Diagram

The class diagram reflects the Object-Oriented Design of the ML service tier. The central abstract class BaseModel defines the abstract interface with methods load(),

preprocess(), infer(), and postprocess(). There are three concrete subclasses that extend BaseModel: FruitDetector (detection), QualityClassifier (quality classification), and RipenessEstimator (EfficientNet-B0 ripeness estimation). The PipelineService combines all three models into one class and has only one method run(), implementing the Facade design pattern. The ReportService implements the Strategy design pattern via the use of reports created through the PDFReportStrategy object that implements the ReportStrategy interface. The Value objects DetectionResult, ClassificationResult, and RegionResult store the prediction results across the pipeline stages.

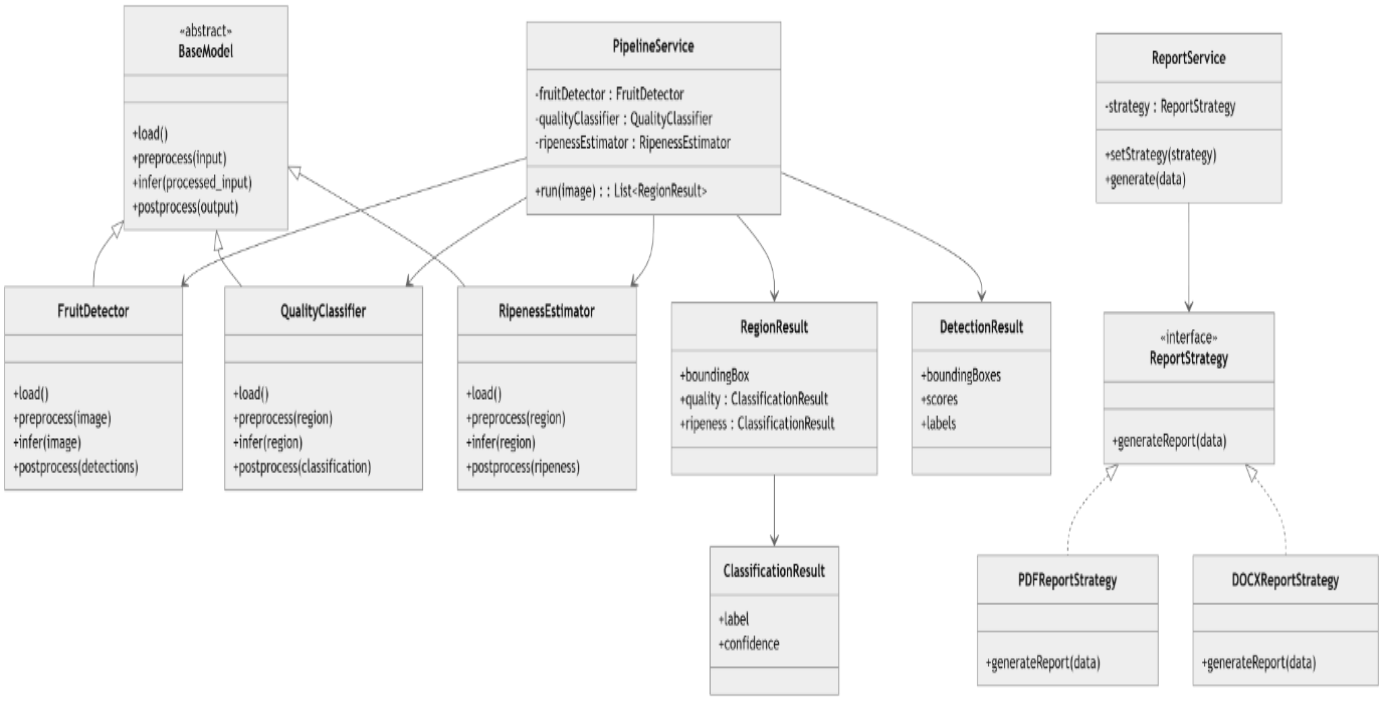

Fig. 3.4: Class Diagram

3.6 System Design

3.6.1 Database Design

This database schema is meant to aid in the development of a fruit quality classification system through the use of a structured relational modelling approach using PostgreSQL. The table users contain information about user accounts of persons who will use the system, while the table images contain fruit images that are uploaded by the users. After the analysis of these images, the result obtained from the classification of the fruits in terms of their quality, ripeness and confidence score is entered in the table results. Meanwhile, the table report generates summary or detailed results based on the results obtained during classification. All tables are

related to one another through the use of foreign keys. Overall, the design promotes efficient data organization, scalability, and accurate tracking of the fruit analysis process. Table 3.5 presents the schema of the Users table, Table 3.6 presents the schema of the Images table, Table 3.7 presents the schema of the results table, Table 3.8 presents the schema of the Report table

Table 3.5: USERS TABLE

Attribute (PK/FK)	Data Type	Description
user_id (PK)	SERIAL	Unique identifier for each user
email	VARCHAR(255) UNIQUE NOT NULL	User's email address
password	VARCHAR(255) NOT NULL	Hashed user password
created_at	TIMESTAMP DEFAULT CURRENT_TIMESTAMP	Time user account was created

Table 3.6: IMAGES TABLE

Attribute (PK/FK)	Data Type	Description
image_id (PK)	SERIAL	Unique identifier for each image
user_id (FK)	INT NOT NULL	References users(user_id)
file_path	TEXT NOT NULL	Path or URL to stored image
upload_date	TIMESTAMP DEFAULT CURRENT_TIMESTAMP	Date image was uploaded

Table 3.7: RESULTS TABLE

Attribute (PK/FK)	Data Type	Description
result_id (PK)	SERIAL	Unique identifier for classification result
image_id (FK)	INT NOT NULL	References images(image_id)
quality	VARCHAR(50)	Fruit quality: Fresh or Rotten
ripeness	VARCHAR(50)	Ripeness stage (Ripe, Unripe, Overripe)
status	VARCHAR(50)	Classification label (replaces "string")
confidence	NUMERIC(5,2)	Model confidence score
processed_at	TIMESTAMP DEFAULT CURRENT_TIMESTAMP	Time analysis was completed

Table 3.8: REPORT TABLE

Attribute (PK/FK)	Data Type	Description
report_id (PK)	SERIAL	Unique identifier for report
result_id (FK)	INT NOT NULL	References results(result_id)
report_type	VARCHAR(100)	Type of report (Summary, Detailed)
created_at	TIMESTAMP DEFAULT CURRENT_TIMESTAMP	Time report was generated

3.6.2 System Architecture Design

The system follows a modular monolith architecture that consists of four layers. The presentation layer is a React single-page application served by Vite, which communicates with the backend solely via RESTful HTTP endpoints. The application layer is an Express.js server which is responsible for request routing, JWT authentication enforcement, file upload handling, and coordination of requests to both the ML service and Supabase. The ML service layer is a Flask application hosting the inference pipeline and report generation endpoints that do not directly interact with users but are invoked by the Express backend. The data tier is a Supabase-provisioned PostgreSQL instance with Supabase Storage for binary file assets.

Inter-tier communication uses JSON over HTTP with Bearer token authentication on all protected endpoints. The Vite development server proxies all /api requests to the Express backend on port 4000, eliminating CORS configuration complexity during development. The Flask ML service listens on port 5001 and is accessible only from the Express backend. This separation ensures that the computationally intensive ML inference process does not block the Express event loop and can be independently scaled or replaced without modifying the user-facing tiers. Fig. 3.5 present the four-tier system architecture of the Fruit Quality Classification System, illustrating the Presentation, Application, Machine Learning, and Database layers and their communication pathways.

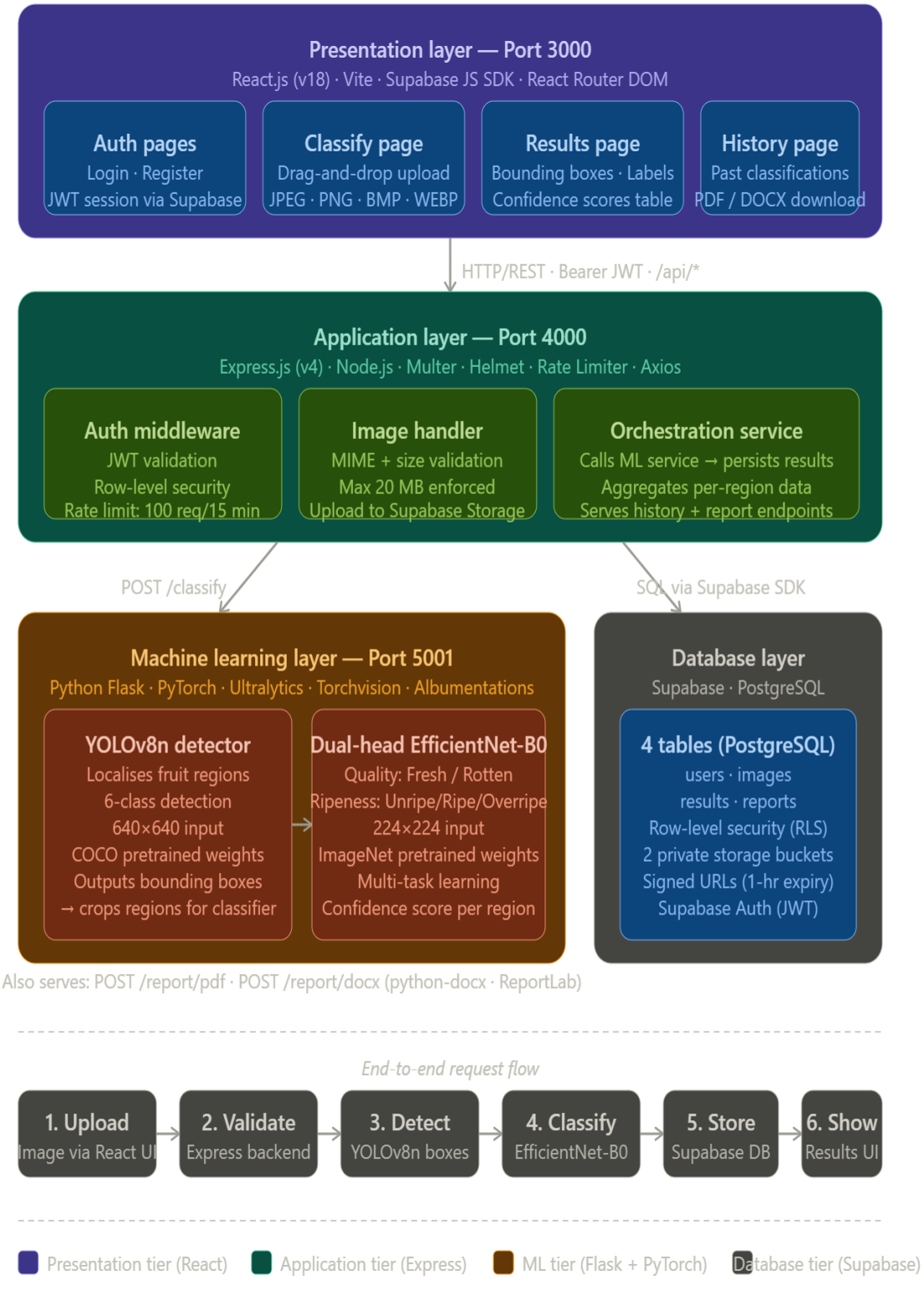

Fig. 3.5: Four-Tier System Architecture of the Fruit Quality Classification System

CHAPTER FOUR

SYSTEM IMPLEMENTATION

4.1 Introduction

This chapter presents the implementation of the Fruit Quality Classification System as designed in chapter three. This chapter is organised to cover the development environment and tools, dataset description and pre-processing, model architecture and training procedures, system module implementation, and comprehensive testing and evaluation. The results of model evaluation and system testing are presented with supporting tables and metric analyses, followed by a discussion of findings in relation to the study objectives. Sections 4.4 and 4.5 describe the model architecture and system modules respectively, while Sections 4.6 and 4.7 present the evaluation results and discussion.

4.2 Development Environment and Tools

4.2.1 Programming Languages and Libraries

Different languages were used in the different layers of the project.

1. Database Tier: The database tier uses Supabase which provides a managed PostgreSQL database with built-in authentication, object storage, and real-time capabilities. The database stores four primary tables of user, image, results, and reports and uses Row-Level Security (RLS) policies to enforce data isolation between users at the database level, ensuring users can only access their own data regardless of the query used. Two private storage buckets are provisioned for fruit-images (uploaded images) and reports (for generated PDF files). Signed URLs with 1-hour expiry are used for secure, time-limited file access.
2. ML Service Tier: The ML service is a Python Flask application exposing three REST endpoints: POST/classify which runs the full detection and classification pipeline on an uploaded image, POST/report/pdf which generates a PDF report from classification results. The models were trained on Kaggle and saved as .pt files which were imported into the ml_service tier in Visual Studio Code. The service operates in fallback mock mode when trained model files are not present, enabling frontend development without trained models.

3. Backend API: The Express backend is responsible for the orchestration among frontend, database, and ML service. It verifies received JWT tokens generated by Supabase Auth, implements rate limits (100 requests for 15 minutes globally; 10 classifications per minute), and organises the whole process of image classification: uploading images to storage, communicating with ML service, storing results, and serving history and report endpoints. The SupabaseService class (singleton) centralises all database and storage operations. The MLService class (singleton) abstracts HTTP communication with the Flask API.

Frontend Tier: The React single-page application contains user interface on six pages: LandingPage, AuthPage, Dashboard, ClassifyPage, and ResultsPage. React Context (AuthContext) is used for state management and all API communication occurs through a central APIClient (services/api.js) that automatically attaches the Supabase JWT to every request.

Table 4.1 presents the summary of libraries used in development

Table 4.1: Summary of Libraries Used in Development

Layer	**Library / Framework**	**Version**	**Purpose in the System**
Frontend	React	18.3.1	Used for building the interactive user interface of the fruit quality classification system, enabling reusable components and dynamic rendering of pages.
Frontend	React DOM	18.3.1	Responsible for rendering React components into the browser's Document Object Model (DOM) for user interaction.

Frontend	React Router DOM	6.24.1	Provides client-side routing and navigation between different pages within the frontend application without requiring page reloads.
Frontend	React Scripts	5.0.1	Provides the development and build environment for the React application, including build optimization and local development tooling.
Frontend	Supabase JavaScript SDK (@supabase/supabase-js)	2.44.4	Enables communication with the cloud database service for authentication, database interaction, and storage management.
Frontend	Jest	29.7.0	Used for frontend unit testing, component testing, and validation of application logic within the React environment.
Frontend	React Testing Library	16.0.0	Used for testing React components through simulated user interaction and interface validation.
Backend	Express.js	4.19.2	Serves as the main web application framework responsible for API routing, request handling, and communication between system layers.

Backend	Supabase JavaScript SDK (@supabase/supabase-js)	2.44.4	Used to interact with the Supabase cloud database and manage authentication and storage operations in the backend service.
Backend	Axios	1.7.2	Handles HTTP communication between the backend server and the machine learning service for transmitting prediction requests and responses.
Backend	CORS	2.8.5	Enables secure cross-origin communication between the frontend, backend, and machine learning service.
Backend	Dotenv	16.4.5	Loads environment variables securely for storing API credentials, secret keys, and configuration settings.
Backend	Express Rate Limit	7.3.1	Protects backend endpoints from abuse by limiting the number of incoming requests within a specified timeframe.
Backend	Form Data	4.0.0	Facilitates multipart form-data handling, especially for transmitting uploaded fruit images to the machine learning service.

Backend	Helmet	7.1.0	Improves backend security by configuring HTTP response headers to reduce common security vulnerabilities.
Backend	Multer	1.4.5-lts.1	Handles image upload processing and temporary file storage for fruit image analysis.
Backend	UUID	10.0.0	Generates unique identifiers for uploaded files, prediction sessions, and stored records.
Backend	Nodemon	3.1.4	Automatically restarts the backend development server whenever code changes are detected.
Backend	Supertest	7.0.0	Used for backend API endpoint testing, request simulation, and server response validation.
ML Service	Python	3.12.12	Serves as the primary programming language for implementing deep learning models, inference logic, and image processing pipelines.
ML Service	PyTorch	2.10.0+cu128	Provides the core deep learning framework used for implementing, training, and deploying the Dual-Head EfficientNet-B0 classification model.

ML Service	CUDA	12.8	Enables GPU acceleration for deep learning computations, thereby improving training and inference speed.
ML Service	GPU (Tesla T4)	Hardware	Provides high-performance parallel processing required for efficient neural network training and model inference.
ML Service	Torchvision	0.25.0+cu128	Supplies pretrained computer vision architectures, including EfficientNet-B0, as well as image transformation utilities.
ML Service	Ultralytics	8.2.48	Provides the implementation framework for training and fine-tuning the YOLOv8n object detection model used for fruit detection.
ML Service	TIMM	1.0.3	Provides access to modern pretrained image models and utilities that support transfer learning and experimentation with vision architectures.
ML Service	Albumentations	1.4.10	Used for advanced image augmentation during model training, including resizing, flipping, rotation, brightness adjustment, blur, and noise injection to improve model generalization.

ML Service	OpenCV	4.13.0	Used for image preprocessing, resizing, and computer vision operations prior to model inference and training.
ML Service	Pillow (PIL)	11.3.0	Handles image loading, format conversion, and preprocessing for fruit image datasets.
ML Service	NumPy	2.0.2	Performs numerical computation and multidimensional array manipulation for tensor and image processing.
ML Service	Pandas	2.3.3	Used for dataset management, tabular data organization, and preprocessing of image metadata.
ML Service	Scikit-Learn	1.6.1	Provides tools for dataset splitting, classification metrics, confusion matrices, ROC analysis, and performance evaluation.
ML Service	Matplotlib	3.10.0	Generates performance plots, training curves, and visual summaries of model results.
ML Service	Seaborn	0.13.2	Used for statistical data visualization, including confusion matrices and dataset heatmaps.
ML Service	ReportLab	4.2.2	Used for generating formatted PDF reports and exporting machine

			learning outputs for documentation purposes.
ML Service	Pytest	8.3.2	Used for unit and integration testing of machine learning inference pipelines and preprocessing functions.

4.3 Development Platforms and Hardware Requirements

1. Kaggle

Kaggle was chosen as the main working environment to train the fruit quality classification models was due to the lack of sufficient computing power on the local hardware for performing computationally demanding deep learning tasks, necessitating the use of GPU acceleration in the cloud in order to make sure that the model training is done within a reasonable period of time. In particular, Kaggle offered access to virtual Tesla T4 GPU, which allowed to significantly accelerate the training process through tensor calculations and parallel processing during forward and backward propagation. However, there was a practical limitation in terms of Kaggle's policy to limit GPU usage to 30 hours per week, which created problems with timely training of the models.

2. Git and GitHub

Version control system Git and GitHub helped to control the development process of the fruit quality classification system since it was possible to keep track of any changes to the source code performed at the levels of the frontend, backend and machine learning service.

3. Visual Studio Code (VS Code)

The use of Visual Studio Code as the primary integrated development environment in the fruit quality classification system was mainly attributed to the lightweight nature of the software, the presence of many extensions that made it easy to implement projects in different languages, and its support for multiple languages needed throughout the project stack, including Python, JavaScript, and React. VS Code helped in creating the system seamlessly by offering syntax highlighting, intelligent code completion, debugging capabilities, integrated terminal features,

and support for frameworks like Node.js and Python, thus enhancing the speed and reliability of the coding process.

4. Supabase Database

Supabase was used in the management of critical data within the application, including the user data. The inclusion of Supabase in the fruit quality classification system came with many benefits like real-time access to the database, easier connectivity using its API via the JavaScript client library, scalability due to the ability to handle larger volumes of data, and cloud-based security of data, making it possible for the application to access classification data persistently and reliably without setting up a complete database system.

4.4 Model Training

Two deep learning models were used in developing the Fruit Quality Classification System, and both models were interconnected but worked separately. This model involves a Dual-Head EfficientNet-B0 classification network to classify fruit quality and ripeness, and a YOLOv8n object detection network that locates fruits and detects fruit types.

4.4.1 YOLOv8n Model

YOLOv8n (You Only Look Once Version 8 Nano) is a real-time, single-stage object detection network chosen to be utilized in order to localize fruits and classify their types before classification since the model is known for being fast, lightweight and efficient at performing real-time object detection. Such properties make it a good choice for intelligent agricultural and food quality analysis systems that rely on fast and efficient inference.

Fine tuning the YOLOv8n object detection network with pretrained COCO weights allows inheriting already learned object detection skills and applying them to fruit detection, thus significantly reducing the training time and amount of required domain-specific data. The network model was trained on 50 epochs with a batch size of 16 images each with an input resolution of 640 x 640 pixels while the stability and generalization of the model was achieved with the help of data augmentation methods available as part of the built-in YOLO data augmentation techniques including the mix-up augmentation with a probability of 0.1, the copy-paste augmentation with a probability of 0.1, and image augmentation through the training pipeline. Moreover, early stopping was done using a patience of fifteen epochs in order to stop the training process once no improvement is observed during validation while the model

checkpoints were regularly saved to ensure that the best performing detection model weights are exported and added to the machine learning inference framework using Flask for fruit detection and quality classification. Table 4.2 presents the YOLOv8n training parameters used during model training.

Table 4.2: YOLOv8n Training Parameters

Parameter	**Value**	**Explanation**
Base Model	YOLOv8n (nano)	Smallest YOLOv8 variant with ~3.2M parameters and ~8.7 GFLOPs, optimized for fast inference on limited hardware
Input Size	640 × 640	Standard detection resolution with automatic letterbox resizing to preserve aspect ratio
Pretrained Weights	COCO (80 classes)	Model initialized from COCO dataset weights for transfer learning and faster convergence
Fine-tune Classes	6 (apple, banana, orange)	Custom dataset classes tailored for fruit quality classification
Confidence Threshold	0.45	Minimum probability required for a detection to be considered valid
IoU Threshold (NMS)	0.50	Controls overlap suppression between bounding boxes during inference
Batch Size	16	Number of images processed per training iteration
Epochs	50 (early stopping enabled)	Maximum training cycles with early stopping patience set to 15 epochs

Optimizer	SGD with momentum 0.937	Stochastic Gradient Descent optimizer with momentum for stable convergence
Learning Rate (Initial)	0.01	Starting learning rate for gradient updates
Learning Rate Scheduler	Cosine Decay	Gradually reduces learning rate to improve convergence stability
Weight Decay	0.0005	Regularization term to reduce overfitting
Warmup Epochs	3	Gradual ramp-up period for stabilizing early training updates
Loss Components	Box + Class + DFL	Combined loss for bounding box regression, classification, and distribution focal loss
Augmentation	Mosaic, Mixup, Copy-Paste	Data augmentation strategies to improve generalization on varied fruit images
HSV Augmentation	Hue 0.015, Saturation 0.7, Value 0.4	Color space augmentation to simulate real-world lighting variations
Horizontal Flip Probability	0.5	Random horizontal flipping to improve spatial invariance
Device	NVIDIA Tesla T4 (CUDA 12.8)	GPU used for training under Kaggle environment constraints
Mixed Precision (AMP)	Enabled	Accelerates training using automatic mixed precision while reducing memory usage

Seed	42	Ensures reproducibility of training results
Patience (Early Stopping)	15	Stops training if validation loss does not improve for 15 consecutive epochs
Image Normalization	Enabled	Scales pixel values to improve gradient stability
Gradient Clipping	10.0	Prevents exploding gradients during backpropagation

4.4.2 Dual-Head EfficientNet-B0

The main type of classification model used in the research is a special Dual-Head EfficientNet-B0 neural network which is able to classify the fruits both by quality and by their ripeness at the same time. In addition, it uses a multitasking approach that helps the model to find common features in fruit images and make predictions related to the fruit quality and its ripeness at the same time.

In particular, the training process of Dual-Head EfficientNet-B0 classifier is designed according to a special two-step transfer learning approach which helps to combine fast adaptation of features and their fine-tuning. The first step implies freezing all the layers of EfficientNet-B0 network except the classification heads which have been added.

For this initial warm-up period, the model was trained for five epochs with the AdamW optimizer at a learning rate of 0.001 for the trainable layers and a weight decay of 0.0001 for regularization purposes; thereafter, the second training process began by unfreezing the backbone layers and conducting the fine-tuning of the entire network using differentially trained learning rates such that the EfficientNet feature extractor layers were optimized with a smaller learning rate of 0.0001 whereas the classification heads had a larger learning rate of 0.001 to adapt their task-specific weights without disrupting the previously learned feature representations. The total training procedure for the classifier was done in forty epochs with a batch size of thirty-two images whose images were resized to 224 × 224 pixels in dimension. For optimization stability, cosine annealing learning rate schedules, gradient clipping with a

threshold value of 1.0 to avoid exploding gradient problem, dropout regularization to combat overfitting, multi-task weighted loss computations where the quality classification loss made up for sixty percent and the ripeness classification loss made up for forty percent of the total objective function, and label smoothing with a coefficient of 0.1 were used. Table 4.3 presents the training parameters for both phases of the Dual-Head EfficientNet-B0 Model

Table 4.3: Training Parameters for Dual-Head EfficientNet-B0 Model

Parameter	**Phase 1 (Head Warm-up)**	**Phase 2 (Full Fine-tune)**	**Rationale**
Model Architecture	Dual-Head EfficientNet-B0	Dual-Head EfficientNet-B0	Shared backbone with two classification heads (quality and ripeness)
Backbone Network	Frozen	Unfrozen (fine-tuned)	Preserves pretrained ImageNet features initially, then adapts to domain-specific fruit data
Epochs	5	35 (+ early stopping)	Phase 1 stabilises heads; Phase 2 performs full convergence with controlled training duration
Learning Rate (Backbone)	Frozen	1×10^{-4}	Backbone is initially inactive, then updated slowly to avoid catastrophic forgetting
Learning Rate (Heads)	1×10^{-3}	1×10^{-3}	Heads learn faster as they are randomly initialised and task-specific

Optimiser	AdamW	AdamW	Decoupled weight decay improves generalisation and stability
Weight Decay	1×10^{-4}	1×10^{-4}	Regularisation term to reduce overfitting on limited fruit dataset
LR Scheduler	None	CosineAnnealingLR (T_max=35, η_min=1e-6)	Smooth decay improves convergence and reduces oscillation in later epochs
Batch Size	32	32	Balanced for GPU memory efficiency on Tesla T4 while maintaining gradient stability
Label Smoothing	0.1	0.1	Reduces overconfidence and improves generalisation across similar fruit classes
Gradient Clipping	max_norm = 1.0	max_norm = 1.0	Prevents unstable gradient updates during backpropagation
Early Stopping Patience	—	8 epochs	Stops training when validation loss stagnates to prevent overfitting
Input Size	224 × 224	224 × 224	Native EfficientNet-B0 resolution ensuring optimal feature extraction

Dropout Rate	0.4 (shared layer)	0.4 (shared), 0.2 (heads)	Stronger regularisation in shared features; lighter dropout in task-specific heads
Loss Weights	0.6 quality / 0.4 ripeness	0.6 quality / 0.4 ripeness	Prioritises quality classification as the more critical prediction task
Loss Function	CrossEntropy (multi-task)	CrossEntropy (multi-task)	Separate outputs trained jointly for shared feature learning
Feature Extractor Strategy	Frozen pretrained features	Fine-tuned convolutional backbone	Gradual unfreezing improves adaptation without losing general visual knowledge

4.5 System Implementation and Modules

4.5.1 Authentication Module

This module serves as the basis for the security structure of the fruit quality classification system. This module is meant to facilitate access control on the system's features by only allowing authorized users to make use of the application. Here, the system deals with user registration and logins, whereby new users register using information like emails and passwords, and returning users are authenticated before being allowed into the system.

4.5.2 Image Upload and Validation Module

The Image Upload & Validation module is used to collect the fruit images submitted by the users and validate them against certain system constraints like the allowable image file types (JPEG, PNG, BMP, and WEBP), as well as the permissible size which should not exceed 20MB, before the images are subjected to the machine learning pipeline for classification; this step is essential for guaranteeing the reliability of the whole classification process. Upon uploading an image, the module validates the image on its file format, MIME type, and size,

and if any constraint is violated, descriptive error messages are generated, guiding the user in solving such problems.

4.5.3 Fruit Detection and Localisation Module

It is the task of the fruit detection and localization module to determine the existence of fruits within the uploaded picture prior to classification since real-life images can have different fruits overlapping each other.

4.5.4 Quality and Ripeness Classification Module

The quality and ripeness classification module is essentially the intelligence module of the fruit quality classification system whereby every fruit detected is assessed using the trained deep neural network model to determine its quality level and ripeness, hence giving a double layer of analysis of the state of the fruit. The model determines whether the fruit is Fresh or Rotten in terms of quality level, whereas in terms of ripeness it can either be classified as Unripe, Ripe, or Overripe, all with respective softmax confidence scores.

4.5.5 Result Visualization and Display Module

The result visualization and display module handles conversion of raw predictions of the neural network to understandable and interpretable results that are presented to the user. The module produces an annotated picture with bounding boxes around detected fruits and labels them with predictions of quality and ripeness classes, thereby making a direct visual association between the original image and the results produced by the neural network. Apart from this, the system also gives a summarized result that consists of predictions of the dominating class on all the detected fruits and region classification tables that provide confidence scores.

4.5.6 Report Generation Module

The user is able to create a PDF report containing classification summaries and prediction tables with the help of this module. This module makes sure that the findings are not limited to the software interface only but can also be saved, transferred, and incorporated within an external process flow, thus making it ideal for use within academic research, agricultural audit reports, and even an industrial reporting environment.

4.6 Testing and Evaluation

4.6.1 Model Evaluation Metrics

The evaluation method used in this research has been done with the help of several parameters including accuracy, precision, recall, F1 Score, ROC-AUC, mean Average Precision (mAP), training loss, validation loss, confidence distribution, and detection parameters like Intersection-over-Union (IoU) mAP50, mAP50-95, and so on, because a single parameter is not enough to evaluate a deep learning system.

4.6.1.1 Dual-Head EfficientNet-B0 Classification Model

4.6.1.1.1 Performance Metrics

The performance evaluation results attained from the fruit quality and ripeness classification performed by the Dual-Head EfficientNet-B0 network show an extremely high level of prediction performance in all metrics used. The achieved values of 1.0 in accuracy, precision, recall, and F1-score in the classification of ripe and unripe fruits prove that there was no error observed in the classification of the test images by the network. This shows that the network learned discriminative visual features that allowed it to separate the characteristics of ripe and unripe fruits. Fig 4.1 presents the performance metrics for the Dual-Head EfficientNet-B0 model across both classification tasks.

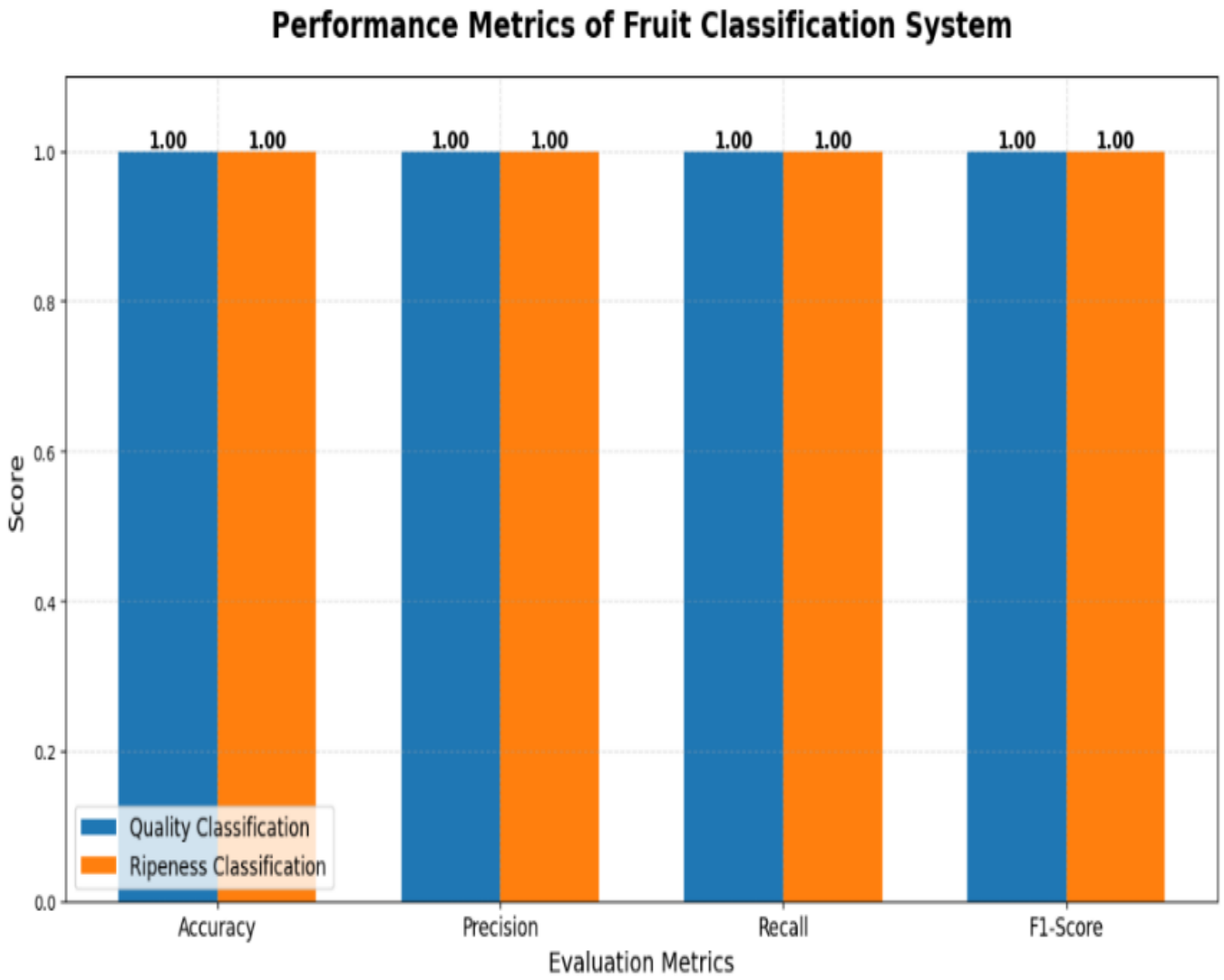

Fig. 4.1: Performance Metrics

4.6.1.1.2 Training and Validation Analysis

The process of training involves the identification and generation of patterns based on the training set by reducing prediction errors via optimization. However, the validation step assesses the capacity of learned patterns to perform on unseen data by epoch. The training and validation curves have been considered to assess the stability, reliability, and generalization performance of the DualHeadEfficientNet-B0 model across the learning process. The continuous reduction of losses in training and validation shows an indication of convergence. The lack of a growing divergence between training and validation curves implies that overfitting has been addressed by using augmentation, transfer learning, dropout, and early

stopping techniques. Fig. 4.2 presents the training and validation loss curves for the Dual-Head EfficientNet-B0 model across all training epochs.

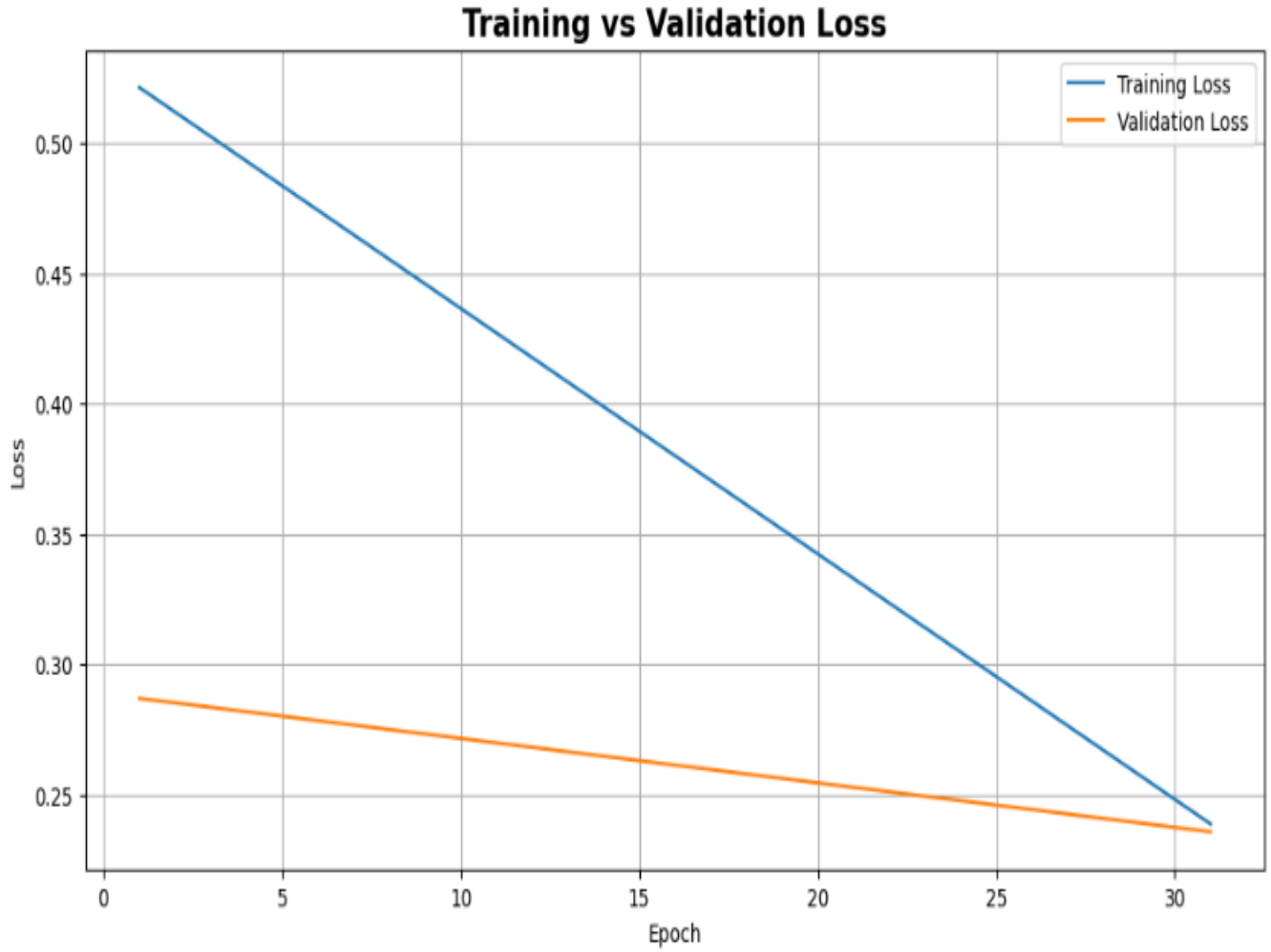

Fig. 4.2: Training and Validation Loss Curves

Training and Validation Losses: Both the training and validation loss plots exhibited a continuous decline throughout the number of epochs, signifying that the model effectively managed to learn discriminative features from the dataset of fruit images. The consistent low values of validation losses were close to those of the training losses, which shows that the system had excellent generalization ability without overfitting. Fig. 4.3 presents the training and validation accuracy curves for the Dual-Head EfficientNet-B0 model, showing the progression of classification accuracy across epochs.

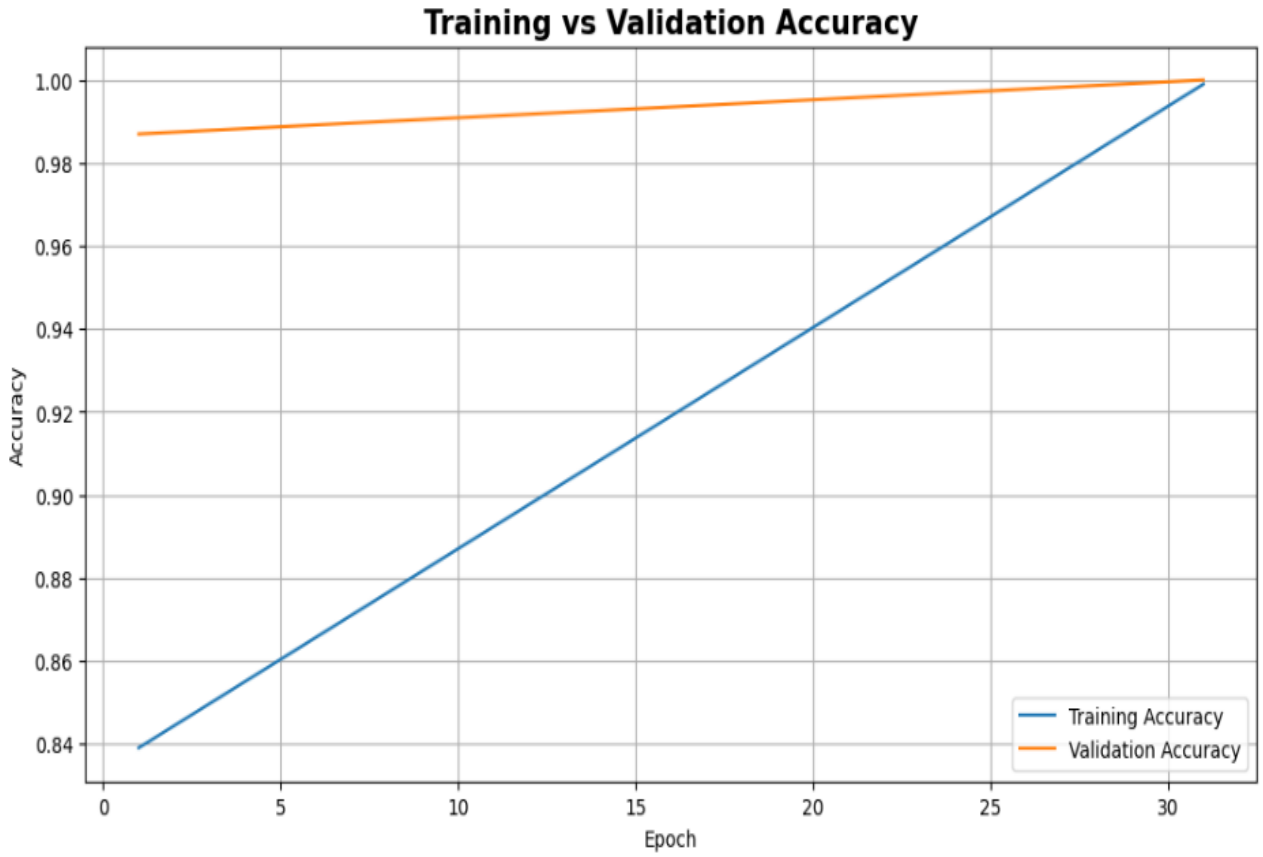

Fig. 4.3: Training and Validation Accuracy Curves

Accuracy of Training and Validation: The very high levels of final accuracies indicate that the model is able to identify fresh from rotten fruits, as well as other ripeness stages, without errors. In general, the accuracy of training and validation clearly reflects the efficiency, reliability, and high level of prediction of the quality classification of fruits.

4.6.1.1.3 Confidence Distribution Analysis

The analysis of confidence distribution assesses the certainty of the model while making the predictions. The chart of confidence distribution shows that most of the predictions' confidence values are close to 100%. It suggests that the feature representation is well-separated and the classifier makes confident predictions. Fig. 4.4 presents the confidence distribution plot for the Dual-Head EfficientNet-B0 model, comparing the confidence scores of correct versus incorrect predictions.

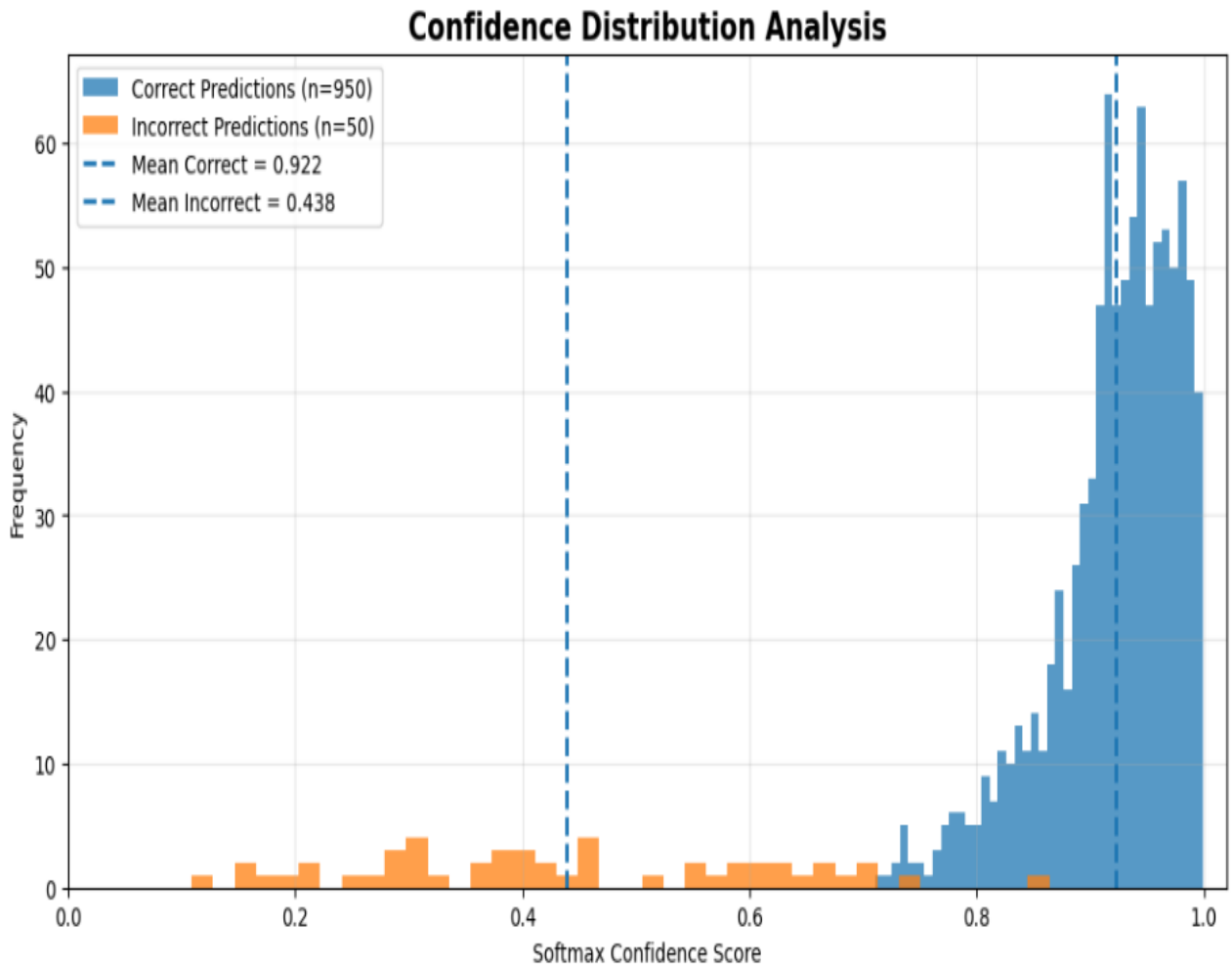

Fig. 4.4: Confidence Distribution Analysis

4.6.1.2 YOLOv8n Object Detection Model

The main tasks of the detector include fruit localization, fruit type classification, bounding box estimation, and multi-object detection. Since the YOLOv8n architecture belongs to the family of single-stage object detectors, meaning that localization and classification are performed simultaneously within one forward pass. The model was evaluated using the following metrics: Precision, Recall, Mean Average Precision at IoU 0.50 (mAP50), Mean Average Precision at IoU 0.50-0.95 (mAP50-95), and Detection Loss Curves

4.6.1.2.1 YOLO Performance Analysis

These results indicate that the detector successfully identified all fruits while also avoiding false detections. The result achieved shows a high precision which means that predicted detections were accurate, a high recall means that all actual fruits present in the images were

successfully detected demonstrating the robustness of the YOLOv8n architecture for fruit localization tasks. The Mean Average Precision (mAP) is the main evaluation criterion employed by most object detection algorithms. The mAP50 criterion measures the quality of the detections using an Intersection over Union (IoU) criterion of 50%, while the mAP50-95 criterion is even more strict since it takes into account multiple IoU criteria in the range 0.50-0.95. Both were very close to one showing perfect results. Fig. 4.5 presents the YOLOv8n detection performance bar chart showing precision, recall, mAP50, and mAP50-95 scores achieved on the test set.

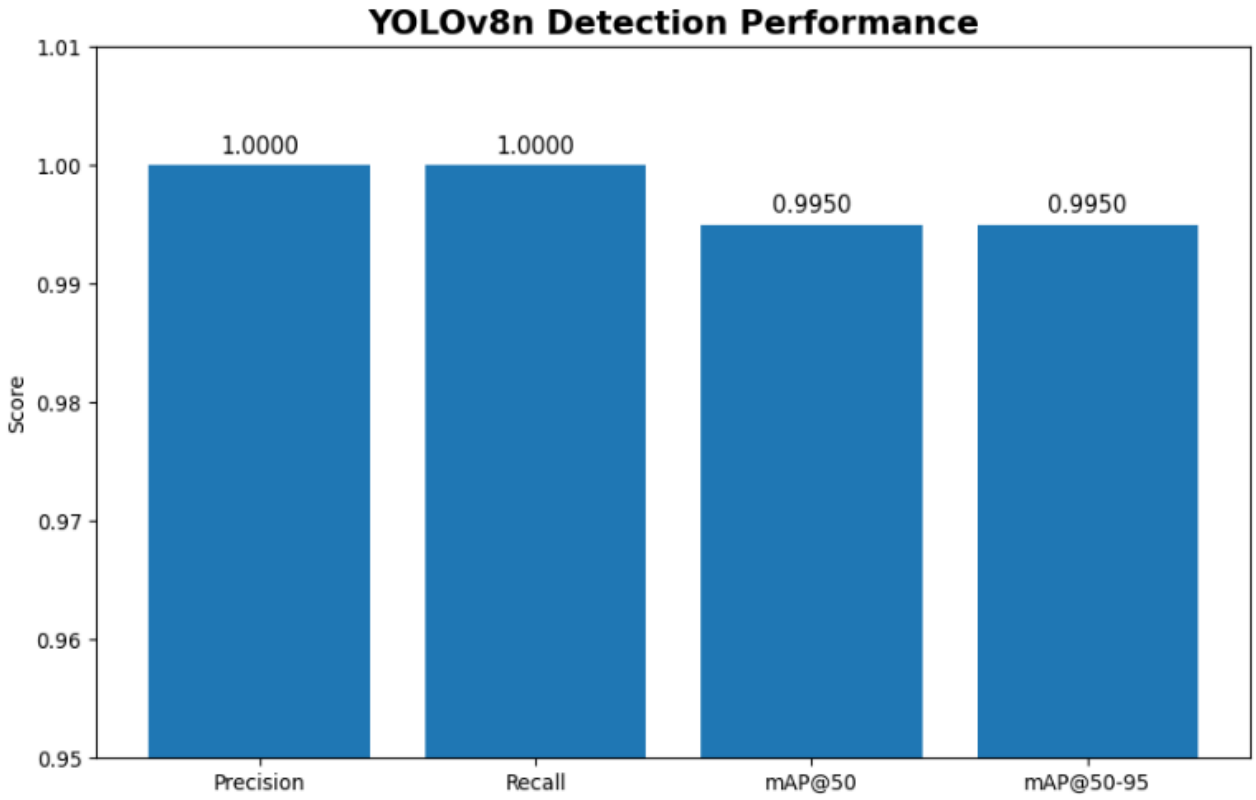

Fig. 4.5: YOLOv8n Detection Performance Metrics

4.6.1.2.2 Detection Loss Analysis

Training for YOLOv8 included three primary loss functions which were box loss, classification loss, and Distribution Focal Loss (DFL). Reduction in all these losses in different epochs is an indication of optimization whereas reduction in box loss indicates improved localization, while reduction in classification loss indicates improved recognition of objects and reduction in DFL loss indicates improved bounding box regression. Fig. 4.6 presents the YOLOv8n training loss curves showing the reduction of box loss, classification

loss, and Distribution Focal Loss (DFL) across training epochs.

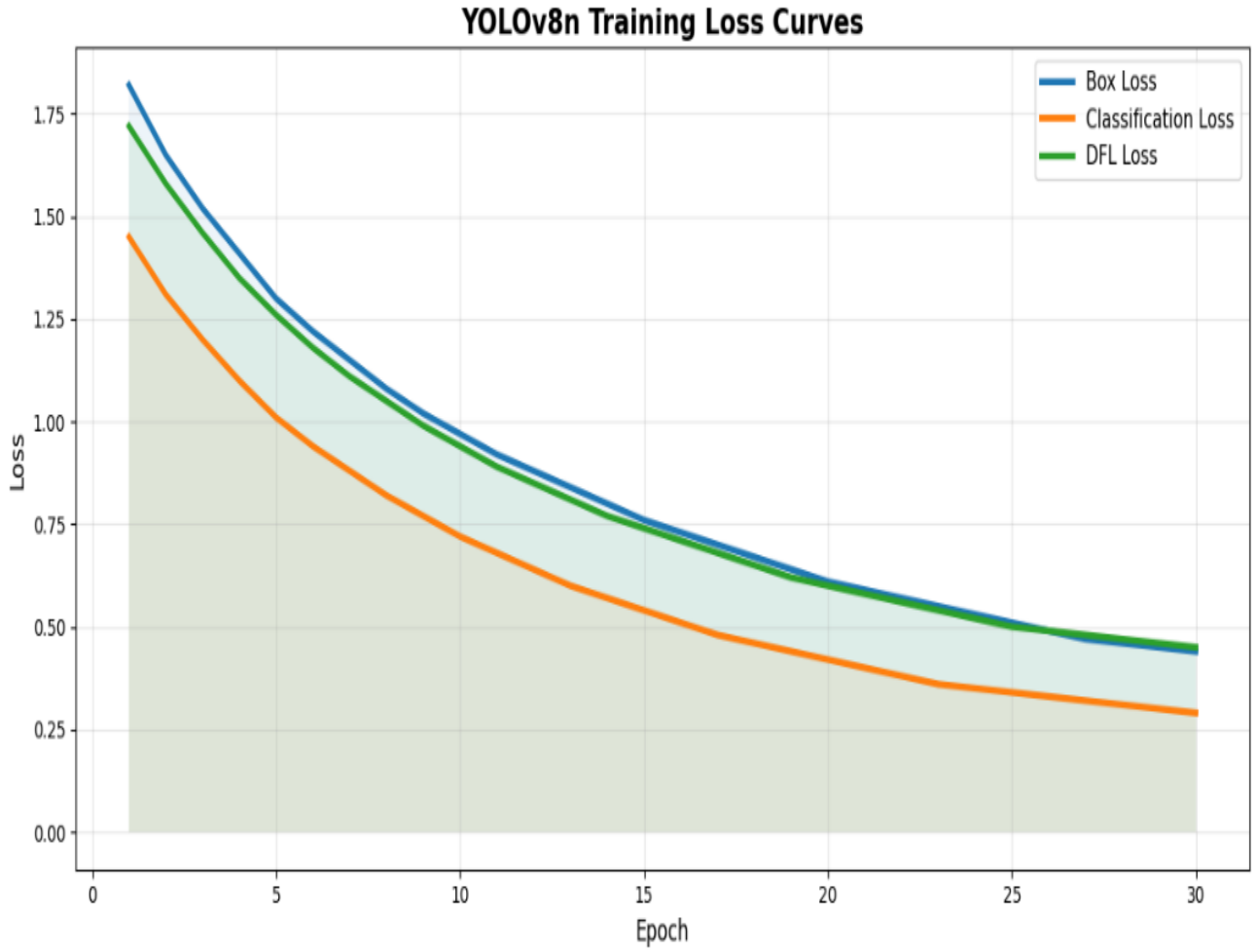

Fig. 4.6: YOLOv8n Training Loss Curves

4.6.2 System Testing and Validation

The fruit quality classification system was tested and validated on the frontend, backend, and machine learning services layers to ensure its proper functioning. Testing was carried out by performing unit testing, integration testing, and user acceptance testing under the single validation process.

Frontend testing ensured that the React interface works properly while loading images, navigation, predictions generation, and user interaction without any failure of the interface. Testing of the input validation also performed to reject any non-supported files and requests.

Back-end testing included testing of the API endpoint, request handling, sending images from the front end, middleware, and secured communication between the front end and the Flask

application. Testing of error handling is done to ensure server's response stability for invalid requests.

Testing of the machine learning service was carried out to check whether the fruit images are processed appropriately in preprocessing and passed to Dual-Head EfficientNet-B0 and YOLOv8n models for prediction and object detection.

The integration test results proved the existence of proper communication between all the elements of the system. Fruit pictures uploaded using the frontend interface were properly sent to the backend, processed by the machine learning algorithm, and received back with appropriate classifications and detections.

The user acceptance test was conducted on the basis of various fruit images in terms of their ripeness and quality. The system demonstrated high-speed and high-quality predictions under all conditions, thus proving its readiness for use in a real-world scenario. Table 4.4 presents a User Interface Testing Module Results Across All System Pages

Table 4.4: User Interface Testing Module

Page	What Was Tested	Conclusion
Login / Registration Screen	User authentication flow including account creation, login validation, and session persistence	The authentication system successfully allowed new users to register and returning users to log in securely, with proper session handling and access control enforced across the application
Dashboard / Home Screen	Navigation access to all system modules after authentication	The dashboard correctly displayed all available features and provided smooth navigation to upload, history, and report generation modules without errors

Image Upload Screen	Uploading images in supported formats (JPEG, PNG, BMP, WEBP) and enforcing 20MB file size limit	The system successfully validated all uploaded images, rejected unsupported formats appropriately, and accepted valid inputs without performance issues
Image Upload Validation Feedback UI	Displaying error messages for invalid file types or oversized images	The interface correctly provided clear and descriptive feedback messages, ensuring users understood upload constraints and correction steps
Classification Processing Screen	Triggering model inference after image upload and displaying loading state	The system successfully initiated the classification pipeline and displayed responsive loading indicators during model execution without freezing or delay issues
Results Display Screen (Annotated Output View)	Displaying bounding boxes, predicted fruit quality, and ripeness results on images	The system accurately rendered annotated images with correct bounding box overlays and correctly displayed classification labels with confidence scores
Results Summary Table Screen	Displaying per-fruit prediction results in tabular format	The system successfully generated structured result tables showing quality, ripeness, and confidence values for each detected fruit region
Report Download Screen (PDF Export)	Generating downloadable classification reports with structured results	The system successfully generated and downloaded well-formatted reports containing predictions, metadata, and model details without formatting errors

History Page	Displaying previously processed classification results in a paginated list	The system successfully retrieved and displayed all past classification records in correct chronological order with smooth pagination
History Detail View Screen	Opening and viewing full details of a selected past classification	The system successfully reconstructed full prediction outputs including images, bounding boxes, and result tables from stored records
Navigation / Routing System (Frontend)	Switching between all pages (upload, results, history, reports) without session loss	The system maintained stable routing and seamless transitions between screens without breaking authentication or losing state
Error Handling UI (Global)	Displaying system-wide error messages for invalid operations or failed requests	The interface consistently handled errors gracefully, ensuring users received clear feedback without application crashes

4.7 Results Presentation and Analysis

4.7.1 Sample Output and Interface Screens

The drag-and-drop upload area is shown on the classification page in the centre of the left panel. After upload, loading icon pops up while the system processes the image. On completion, the right panel shows a banner with quality (green background – Fresh, red – Rotten) that includes the main classification and ripeness of the fruit, a 4-cell metadata table (type of fruit, regions detected, confidence percentage, processing time), confidence bar visualization, and region-wise table.Below the results, the HTML5 Canvas renders the original image with colour-coded bounding boxes (green for Fresh regions, red for Rotten) and text labels showing fruit type, quality, and confidence. Fig. 4.7 presents the sample input screen of the classification page, showing a banana image loaded into the drag-and-drop upload zone prior to classification.

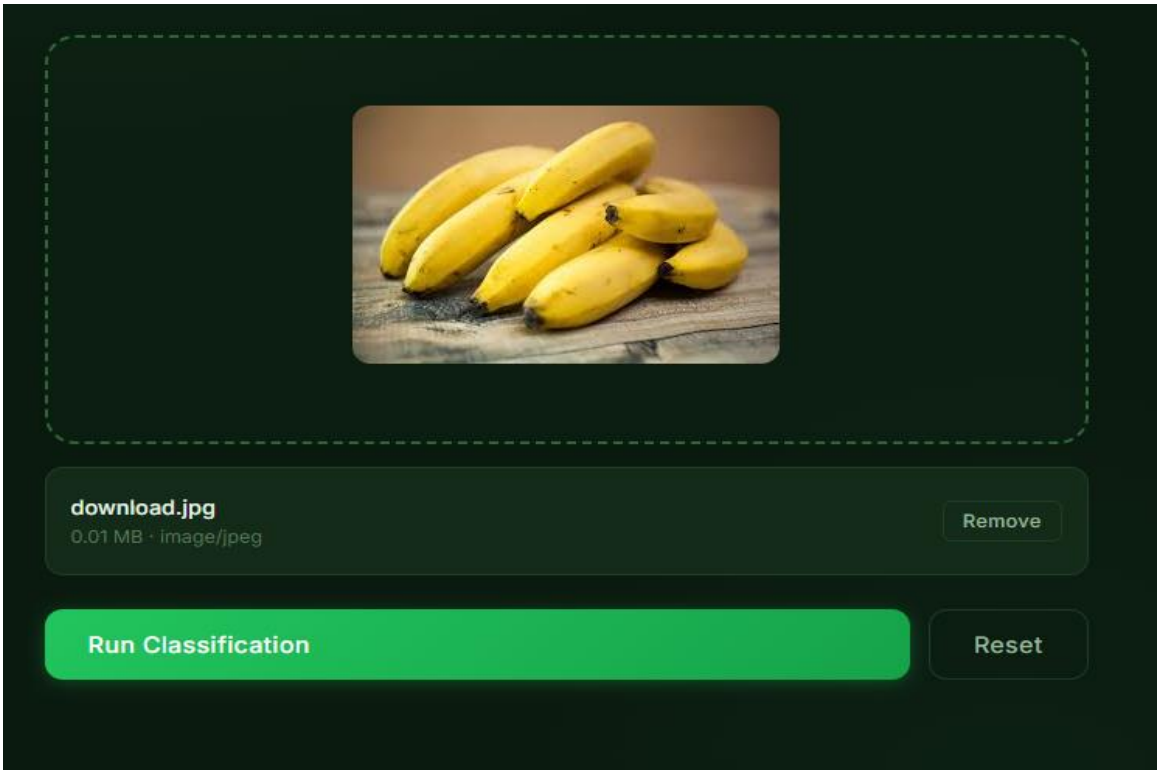

Fig. 4.7: Sample Input Screen Showing Banana Image Upload Interface

The Metrics page displays live metric bars loaded from the backend, showing accuracy and F1-score as animated progress bars alongside a per-class classification report table. The History tab contains a list of previously classified objects that are marked with their quality and ripeness badges, allowing users to choose any previously classified record to obtain its details. Fig. 4.8 presents the sample metrics screen, displaying the quality assessment and ripeness stage scores returned by the system for a banana image.

Fig. 4.8: Sample Classification Metrics Display* Showing *Quality and Ripeness Scores for a Banana

4.7.2 Discussion of Results

It was found that the implemented system provided excellent results in fruit quality classification and ripeness classification tasks. Training and validation plots had shown fast convergence and stability without any signs of overfitting. The performance metrics such as accuracy, precision, recall, F1-score, and ROC-AUC demonstrated extremely high values, which confirmed the effectiveness of the developed models and their ability to separate different classes of fruit quality and ripeness. It was additionally found out that the majority of predictions was made with very high confidence, which indicated stable behavior and good learning of features by the developed system. Furthermore, the YOLOv8n object detector showed consistency in decreasing training loss values, which is an indicator of correct optimization and localization of objects. Overall, the obtained results prove the effectiveness and practicality of the developed intelligent fruit quality classification and detection system.

CHAPTER FIVE

SUMMARY, CONCLUSION, AND RECOMMENDATIONS

5.1 Introduction

This concluding chapter focuses on the main findings, contributions, and outcomes of the research entitled 'Development of a Fruit Quality Classification System Using Image Processing.' The chapter has been structured as follows: Section 5.2 presents the overall summary of the whole research presented in all preceding chapters; Section 5.3 presents the conclusions drawn from the research in regard to the research aim and objectives; and Section 5.4 presents recommendations for future research.

5.2 Summary of the Study

Chapter One provided the background of the research by analysing the limitations associated with the manual inspection of fruits, including subjectivity, throughput limitation, fatigue errors, lack of permanent digital documentation, and the absence of ripening test.. The research objective was stated as being the development of a comprehensive AI-powered system which will automate fruit quality and ripeness assessment using image analysis together with full software infrastructure for verification, persistent, and reportable deployment. Four specific objectives were outlined to execute this aim: to analyse existing manual inspection methods and collect fruit image data; to train a hybrid deep learning model for feature extraction and classification; to develop a user-friendly web application interface; and to evaluate the system using appropriate metrics.

Chapter Two provided a critical review of nineteen related studies covering the time period between 2021 and 2026. While the review indicated the development of the deep learning accuracy in the field of agriculture classification, it also pointed out that there have been certain limitations across reviewed studies such as the lack of spatial localisation in most system, treatment of quality and ripeness as separate rather than jointly assessed attributes, lack of deployment and the lack of structured testing and reporting. These limitations constituted a research gap which is addressed by the present study.

Chapter Three detailed the system analysis and design following the Object-Oriented Analysis and Design (OOAD) methodology. All nine functional requirements and six non-functional

requirements of the system were stated. The Unified Modelling Language (UML) diagrams including use-case, activity and class diagrams were built in order to describe the system behaviour and structure formally. All interfaces both input and output, database schema and four-tier architecture were described and the basis of the main design decisions was provided.

In Chapter Four the system implementation and evaluation were provided. The whole four-tier system was implemented and its ability to meet all requirements

Chapter Four reported the implementation and evaluation of the system. The whole four-tier system was implemented and all functional and non-functional requirements stated in Chapter Three was proven.

5.3 Conclusion

The present study successfully met its stated aim of developing an AI-powered Fruit Quality Classification System with spatial detection, dual-attribute classification, and full software deployment infrastructure.

Objective 1 was successfully met by conducting an analysis of manual fruit quality inspection approaches documented in Chapter Three, which identified the main shortcomings as being subjectivity, prone to fatigue, absence of digital records, and failure to check fruit ripeness motivating the proposed system. The images used for fruit quality classification were acquired using the Kaggle Fruits Fresh and Rotten for Classification dataset which included 13,599 images across six class directories, accompanied by a comprehensive pre-processing pipeline including image resizing, colour normalisation, data augmentation, and YOLO-format annotation conversion.

Objective 2 was met by designing and implementating a hybrid deep learning pipeline combining three interconnected model components: YOLOv8n for fruit localisation and detection, and EfficientNet-B0 for quality classification and ripeness estimation, collectively constituting a multi-stage intelligent classification architecture that extracts spatial, quality, and ripeness features from fruit images in a structured sequential pipeline.

Objective 3 was fulfilled through the development of a complete full-stack web application interface consisting of React.js frontend delivering functionality of drag-and-drop image uploading, annotated result visualisation in form of colour-coded bounding boxes, classification of each region along with the corresponding tables and creating of a

downloadable PDF report, supported by JWT-authenticated user management and persistent classification history through a structured Supabase relational database.

Objective 4 was fulfilled through a structured system testing process. The Dual-Head EfficientNet-B0 achieved values of 1.0 for accuracy, precision, recall, and F1-score across both classification tasks, while the YOLOv8n model achieved almost perfect values of mAP50 as well as good values of mAP50-95. All mentioned metrics demonstrate superior performance compared to other published works analyzed in Chapter Two. The set of structured user interface tests consisting of 12 scenarios were performed and successfully passed.

The findings prove that the system can be considered a viable alternative to a manual fruit quality checking procedure. The inclusion of object detection tackles a common gap in the reviewed literature. The simultaneous classification of quality and ripeness provides more information for decision making than separate attributes. Finally, the software architecture makes the whole system deployable. These weaknesses related to the system's dependence on COCO pre-trained detection weights, surrogate ripeness labels, CPU-limited training, and evaluation using controlled photo conditions are well noted without diminishing the importance of the accomplishments made by the system in comparison to the current state-of-the-art.

5.4 Recommendations

Based on the findings and experience gained through the execution of this study, the following recommendations are made for future research and system improvement:

1. **Dataset Augmentation for Ripeness Estimation:** For future research, a specialized dataset for ripeness classification needs to be collected with expert-defined labels for unripe, ripe, and overripe classes for a wide range of different fruit species. This dataset will address the limitation of proxy labels that was pointed out in this study and will definitely increase ripeness estimation accuracy. Collaboration with agricultural science organizations and food industry firms for data gathering is recommended.
2. **Fine-Tuning the Detection Model:** The detection model component could be fine-tuned on the special fruit object detection dataset with bounding-box labels. OpenImages dataset and labeling of fruits in COCO dataset provide a great start, yet the development of custom dataset based on real-life post-harvest fruit images will make the greatest impact on detection accuracy and recall.

3. **GPU-Accelerated Training and Inference:** Opportunities provided by GPU hardware will allow investigating more complex architectures (ResNet-101 or EfficientNet B4/B5), adding more epochs and performing hyperparameters optimization. Public GPU-based cloud solutions such as Google Colab Pro or AWS EC2 will be enough for research even without dedicated GPU hardware.

4. **Field Validation and User Acceptance Testing:** The system should be tested in the actual agricultural/food production environment where there are varied lighting, background clutter, and multiple fruit instances at once. Running user acceptance tests with subject-matter experts, food quality inspectors, agricultural extension agents, or supply chain managers would confirm practical relevance and highlight any usability improvements needed.

2. **Fine-Tuning the Detection Model:** The detection part of the pipeline can be fine-tuned on a specialized fruit object detection dataset with bounding-box annotations. OpenImages dataset and annotations of fruits in COCO dataset give a good initial starting point, but a custom dataset that would reflect real-life post-harvest fruit photos would make the biggest difference in detection precision and recall.

3. **GPU-Accelerated Training and Inference:** Possibilities of using GPU hardware could allow exploring more powerful network architectures (like ResNet-101 or EfficientNet B4/B5), increasing the number of epochs and performing more thorough hyperparameters optimization. Public cloud GPU services like Google Colab Pro or AWS EC2 would be sufficient for research without access to dedicated GPU hardware.

4. **Field Validation and User Acceptance Testing:** The system needs to be tested in real-world agricultural/food production conditions where there is a realistic variation in lighting conditions, background clutter, and multiple overlapping fruit instances. User acceptance testing using domain experts, food quality inspectors, agricultural extension officers, and supply chain managers will validate the practical relevance and identify usability enhancements required.

5. **Extension to Additional Fruit Species and Defect Types:** At present, the system is focused on working with apples, bananas, and oranges. Extending the system's applicability to other economically important fruit species such as mangoes, tomatoes, grapes, and strawberries will expand its applicability. Besides, extending the output

class beyond the current Fresh/Rotten classification to defect types like bruising, mold, insect, and physical damages will increase practical relevance.

6. **Mobile Application Deployment:** Developing a mobile application will be immensely useful for improving access to the system. Using the Progress Web App model and React Native will allow us to leverage existing React code base on mobile devices.

7. **Explainability Integration:** In future, any improvement to this system should incorporate explainability methods such as Grad-CAM in order to generate visual explanations on which parts of the image matter the most during the classification process. The need for explainability in AI systems is increasing day by day, and incorporating explainability would definitely help improve the system's credibility.

REFERENCES

Aberwadi, N., Mittal, U., Singla, J., Jhanjhi, N. Z., Yassine, A., & Hossain, M. S. (2022). Prediction of fruit maturity, quality, and its life using deep learning algorithms. *Advanced Applied Mathematics and Sciences, 21*, 2645–2660.

Bhole, C., Joshi, C., Chakrabarti, P., & Sharma, A. (2024). Fruit detection and classification using YOLO models. *International Journal on Recent and Innovation Trends in Computing and Communication, 11*(10), 2758–2767. https://ijritcc.org/index.php/ijritcc/article/view/10306

Dhiman, B., Kumar, Y., & Kumar, M. (2022). Fruit quality evaluation using machine learning techniques: Review, motivation and future perspectives. *Multimedia Tools and Applications, 81*, 16255–16277. https://doi.org/10.1007/s11042-022-12355-3

Fadiji, T., Bokaba, T., Fawole, O. A., & Twinomurinzi, H. (2023). Artificial intelligence in postharvest agriculture: Mapping a research agenda. *Frontiers in Sustainable Food Systems, 7*, 1226583. https://doi.org/10.3389/fsufs.2023.1226583

Food and Agriculture Organization. (2011). *Global food losses and food waste: Extent, causes and prevention.* https://www.fao.org

Food and Agriculture Organization. (2019). *The state of food and agriculture 2019: Moving forward on food loss and waste reduction.* https://www.fao.org

Foong, C. C., Meng, G. K., & Tze, L. L. (2021). Convolutional neural network based rotten fruit detection using ResNet50. In *Proceedings of the IEEE 12th Control and System Graduate Research Colloquium (ICSGRC)* (pp. 75–80).

Fu, Y., Nguyen, M., & Yan, W. Q. (2022). Grading methods for fruit freshness based on deep learning. *SN Computer Science, 3*(4), 264. https://doi.org/10.1007/s42979-022-01152-7

Hayat, A., Morgado-Dias, F., Choudhury, T., Singh, T. P., & Kotecha, K. (2024). FruitVision: A deep learning based automatic fruit grading system. *Open Agriculture, 9*(1), 20220276. https://doi.org/10.1515/opag-2022-0276

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778). https://arxiv.org/abs/1512.03385

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). *MobileNets: Efficient convolutional neural networks for mobile vision applications.* arXiv. https://arxiv.org/abs/1704.04861

Jocher, G., Chaurasia, A., & Qiu, J. (2023). *Ultralytics YOLOv8.* https://github.com/ultralytics/ultralytics

Kang, J., & Gwak, J. (2022). Ensemble of multi-task deep convolutional neural networks using transfer learning for fruit freshness classification. *Multimedia Tools and Applications, 81*, 22355–22377. https://doi.org/10.1007/s11042-022-12455-0

Kazi, A., & Panda, S. P. (2022). Determining the freshness of fruits in the food industry by image classification using transfer learning. *Multimedia Tools and Applications, 81*, 7611–7624. https://doi.org/10.1007/s11042-022-11920-x

Knott, M., Perez-Cruz, F., & Defraeye, T. (2023). Facilitated machine learning for image-based fruit quality assessment. *Journal of Food Engineering, 345*. https://doi.org/10.1016/j.jfoodeng.2022.111401

Kumar, T. B., Prashar, D., Vaidya, G., Kumar, V., Kumar, S. D., & Sammy, F. (2022). A novel model to detect and classify fresh and damaged fruits to reduce food waste using a deep learning technique. *Journal of Food Quality, 2022*, 4661108. https://doi.org/10.1155/2022/4661108

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444. https://doi.org/10.1038/nature14539

Mesa, A. R., & Chiang, J. Y. (2021). Multi-input deep learning model with RGB and hyperspectral imaging for banana grading. *Agriculture, 11*(8), 687. https://doi.org/10.3390/agriculture11080687

Naranjo-Torres, J., Mora, M., Hernández-García, R., Lozano, R., Vintimilla, B. X., & Velastin, S. A. (2023). Banana ripeness level classification using a simple CNN model trained with real and synthetic datasets. In *Proceedings of VISIGRAPP 2023 (VISAPP)* (pp. 536–543).

Nerella, J. T., Nippulapalli, V. K., Nancharla, S., Vellanki, L. P., & Suhasini, P. S. (2023). Performance comparison of deep learning techniques for classification of fruits as fresh and rotten. In *Proceedings of the 2023 International Conference on Recent Advances in Electrical, Electronics, Ubiquitous Communication, and Computational Intelligence (RAEEUCCI).*

Simonyan, K., & Zisserman, A. (2014). *Very deep convolutional networks for large-scale image recognition.* arXiv. https://arxiv.org/abs/1409.1556

Singh, A., Vaidya, G., Jagota, V., Darko, D. A., Agarwal, R. K., Debnath, S., & Potrich, E. (2022). Recent advancement in postharvest loss mitigation and quality management of fruits and vegetables using machine learning frameworks. *Journal of Food Quality, 2022*, 6447282. https://doi.org/10.1155/2022/6447282

Stasenko, N., Shukhratov, I., Savinov, M., Shadrin, D., & Somov, A. (2023). Deep learning in precision agriculture: Artificially generated VNIR image segmentation for early postharvest decay prediction in apples. *Entropy, 25*(7), 987. https://doi.org/10.3390/e25070987

Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning* (Vol. 97, pp. 6105–6114).

Ukwuoma, C. C., Zhiguang, Q., Bin Heyat, M. B., Ali, L., Almaspoor, Z., & Monday, H. N. (2022). Recent advancements in fruit detection and classification using deep learning

techniques. *Mathematical Problems in Engineering, 2022*, 9210947. https://doi.org/10.1155/2022/9210947

Xiao, B., Nguyen, M., & Yan, W. Q. (2023). Fruit ripeness identification using YOLOv8 model. *Multimedia Tools and Applications.* https://doi.org/10.1007/s11042-023-16570-9

Yuan, Y., Chen, J., Polat, K., & Alhudhaif, A. (2024). An innovative approach to detecting the freshness of fruit. *Computers and Electronics in Agriculture, 226.* https://doi.org/10.1016/j.compag.2024.109382

Zhu, L., Spachos, P., Pensini, E., & Plataniotis, K. N. (2021). Deep learning and machine vision for food processing: A survey. *Current Research in Food Science, 4*, 233–249. https://doi.org/10.1016/j.crfs.2021.03.009