Deep Learning Approach Using ConvNeXt- Tiny and EfficientNetV2-B0 for Multy-Fruit Quality prediction

A Project Report submittedinthe partial fulfillment of the Requirements for the award of the degree

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING
Submitted by

Kambhampati John Wesly	**(22471A0598)**
Bokkisam Naga Anil Kumar	**(22471A0579)**
Vinnakota Manoj Kumar	**(22471A05D8)**
Shaikthettu Sharif	**(22471A05D4)**

Under the esteemed guidance of

Abburi Ramesh, MTech..

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NARASARAOPETAENGINEERING COLLEGE: NARASAROPET

(AUTONOMOUS)

Accredited by NAAC with A+ Grade and NBA under

Tyre -1 an ISO 9001:2015 Certified

Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada

KOTAPPAKONDAROAD, YALAMANDA VILLAGE, NARASARAOPET- 522601

2025-2026

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project that is entitled with the name Deep Learning Approach Using ConvNeXt-Tiny and EfficientNetV2-B0 for Multi-Fruit Quality Prediction **is a bonafide work done by the team Kambhampati John Wesly (22471A0598), Bokkisam Naga Anil Kumar (22471A0579), Vinnakota Manoj Kumar (22471A05D8), Shaikthettu Sharif (22471A05D4)** BACHELOR OF TECHNOLOGY in the Department of **COMPUTER SCIENCE AND ENGINEERING** during **2025-2026.**

PROJECT GUIDE
Abburi Ramesh, MTech.
Assistant Professor

PROJECT CO-ORDINATOR
Dr. Sireesha Moturi, B.Tech., M.Tech., Ph.D
Associate Professor

HEAD OFTHE DEPARTMENT
Dr. S. N. Tirumala Rao, M.Tech., Ph.D.
Professor& HOD

EXTERNAL EXAMINER

DECLARATION

We declare that this project work titled "Deep Learning Approach Using ConvNeXt-Tiny and EfficientNetV2-B0 for Multi-Fruit Quality Prediction" is composed by ourselves that the work contain here is our own except where explicitly stated otherwise in the text and that this work has been not submitted for any other degree or professional qualification except as specified.

Kambhampati John Wesly	(22471A0598)
Bokkisam Naga Anil Kumar	(22471A0579)
Vinnakota Manoj Kumar	(22471A05D8)
Shaikthettu Sharif	(22471A05D4)

ACKNOWLEDGEMENT

We wish to express my thanks to carious personalities who are responsible for the completion of the project. We are extremely thankful to our beloved chairman sri **M. V. Koteswara Rao**, **B.Sc.**, who took keen interest in us in every effort throughout thiscourse. We owe out sincere gratitude to our beloved principal **Dr. S. Venkateswarlu, Ph.D.**, for showing his kind attention and valuable guidance throughout the course.

We express our deep felt gratitude towards **Dr. S. N. Tirumala Rao, M.Tech., Ph.D.,** HOD of CSE department and also to our guide **Abburi Ramesh**, **M.Tech.**, whose valuable guidance and unstinting encouragement enable us to accomplish our project successfully in time.

We extend our sincere thanks towards **Dr. Sireesha Moturi, B.Tech, M.Tech.,Ph.D.,** Associate professor & Project coordinator of the project for extending her encouragement. Her profound knowledge and willingness have been a constant source of inspiration for us throughout this project work.

We extend our sincere thanks to all other teaching and non-teaching staff to department for their cooperation and encouragement during our B.Tech degree.

We have no words to acknowledge the warm affection, constant inspiration and encouragement that we received from our parents.

We affectionately acknowledge the encouragement received from our friends and those who involved in giving valuable suggestions had clarifying out doubts which had really helped us in successfully completing our project.

By

Kambhampati John Wesly	(22471A0598)
Bokkisam Naga Anil Kumar	(22471A0579)
Vinnakota Manoj Kumar	(22471A05D8)
Shaikthettu Sharif	(22471A05D4)

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

INSTITUTE VISION AND MISSION

INSTITUTION VISION

To emerge as a Centre of excellence in technical education with a blend of effective student centric teaching learning practices as well as research for the transformation of lives and community,

INSTITUTION MISSION

M1: Provide the best class infra-structure to explore the field of engineering and research

M2: Build a passionate and a determined team of faculty with student centric teaching, imbibing experiential, innovative skills

M3: Imbibe lifelong learning skills, entrepreneurial skills and ethical values in students for addressing societal problems

NARASARAOPETA ENGINEERING COLLEGE (AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION OF THE DEPARTMENT

To become a centre of excellence in nurturing the quality Computer Science & Engineering professionals embedded with software knowledge, aptitude for research and ethical values to cater to the needs of industry and society.

MISSION OF THE DEPARTMENT

The department of Computer Science and Engineering is committed to

M1: Mould the students to become Software Professionals, Researchers and Entrepreneurs by providing advanced laboratories.

M2: Impart high quality professional training to get expertize in modern software tools and technologies to cater to the real time requirements of the Industry.

M3: Inculcate team work and lifelong learning among students with a sense of societal and ethical responsibilities.

NEC NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Specific Outcomes (PSO's)

PSO1: Apply mathematical and scientific skills in numerous areas of Computer Science and Engineering to design and develop software-based systems.

PSO2: Acquaint module knowledge on emerging trends of the modern era in Computer Science and Engineering

PSO3: Promote novel applications that meet the needs of entrepreneur, environmental and social issues.

Program Educational Objectives (PEO's)

The graduates of the programme are able to:

PEO1: Apply the knowledge of Mathematics, Science and Engineering fundamentals to identify and solve Computer Science and Engineering problems.

PEO2: Use various software tools and technologies to solve problems related to academia, industry and society.

PEO3: Work with ethical and moral values in the multi-disciplinary teams and can communicate effectively among team members with continuous learning.

PEO4: Pursue higher studies and develop their career in software industry.

NEC NARASARAOPETA ENGINEERING COLLEGE (AUTONOMOUS)

Program Outcomes

PO1: Engineering Knowledge: Apply knowledge of mathematics, natural science,computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems.

PO2: Problem Analysis: Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)

PO3: Design/Development of Solutions: Design creative solutions for complex-engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5)

PO4: Conduct Investigations of Complex Problems: Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).

PO5: Engineering Tool Usage: Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)

PO6: The Engineer and The World: Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).

PO7: Ethics: Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)

PO8: Individual and Collaborative Team work: Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

PO9: Communication: Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences

PO10: Project Management and Finance: Apply knowledge and understanding of engineering management principles and economic decision- making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.

PO11: Life-Long Learning: Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change.

Project Course Outcomes (CO'S):

CO421.1: Analyse the SystemofExaminations and identifythe problem.

CO421.2: Identifyand classifythe requirements.

CO421.3: Review the Related Literature

CO421.4: Design and Modularize the project

CO421.5: Construct, Integrate, Test and Implement the Project.

CO421.6: Preparethe project Documentation and present the Report using appropriate method.

Course Outcomes – Program Outcomes mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2	PSO3
C421.1		✓										✓		
C421.2	✓		✓		✓							✓		
C421.3				✓		✓	✓	✓				✓		
C421.4			✓			✓	✓	✓				✓	✓	
C421.5					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C421.6									✓	✓	✓	✓	✓	

Course Outcomes – Program Outcome correlation

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2	PSO3
C421.	2	3										2		
C421.			2		3							2		
C421.				2		2	3	3				2		
C421.			2			1	1	2				3	2	
C421.					3	3	3	2	3	2	2	3	2	1
C421.									3	2	1	2	3	

Note: The values in the above table represent thelevel of correlation between CO's and PO's:

1. **Low level**

2. **Medium level**

3. **High level**

Project mapping with various courses of Curriculum with Attained PO's:

Name of the course from which principles are applie in this project	**Description of the device**	**Attained PO**
C2204.2, C22L3.2	Gathering the requirements and defining the problem, plan to develop model for detection and classification of OSCC	PO1, PO3,PO8
CC421.1, C2204.3, C22L3.2	Each and every requirement critically analyzed, the process mode is identified	PO2, PO3, PO8
CC421.2, C2204.2, C22L3.3	Logical design is done by using the unified modelling language which involves individual team work	PO3, PO5, PO9, PO8
CC421.3, C2204.3, C22L3.2	Each and every module is tested, integrated, and evaluated in our project	PO1, PO5, PO8
CC421.4, C2204.4, C22L3.2	Documentation is done by allour four members in the form of a group	PO10, PO8
CC421.5, C2204.2, C22L3.3	Each and every phase of the work in group is presented periodically	PO8,PO10, PO11
C2202.2, C2203.3, C1206.3, C3204.3, C4110.2	Implementation is done and the project will be handled by the social media users and in future updates in our project can be done based on detection for Oral Cancer	PO4, PO7, PO8
C32SC4.3	The physical design includes website to check OSCC	PO5, PO6, PO8

ABSTRACT

Automating fruit sorting is difficult because produce varies so much in appearance and datasets are rarely balanced. Existing solutions tend to focus on solving each of these problems, which affects the overall efficiency of the system.This paper proposes a dual-stage framework based deep learning system for the integration of these two tasks. For classification of six fruit categories in the FruitNet dataset, a ConvNeXt-Tiny model pretrained on ImageNet is used, and an EfficientNetV2-B0 model that has undergone same-domain transfer learning is used for quality level grading (good, average, poor).Standard geometric augmentations and class-weighted loss functions are applied to mitigate class imbalance and enhance robustness.With a solid F1-score of 95.6%, the system proves reliable enough for real world use on edge devices, the proposed system is superior to the existing approaches.Mostly, the suggested modular and lightweight architecture showcases the practicality of instant fruit evaluation on underpowered gadgets as a first important move for extending agricultural automation.

INDEX

1. INTRODUCTION	**1**
1.1 Motivation	4
1.2 Problem Statement	5
1.3 Objectives	5
2. LITERATURE SURVEY	**7**
2.1 Deep Learning in Agricultural Image Analysis	7
2.2 CNN-Based Approaches for Fruit Classification and Quality Grading	8
2.3 Hybrid CNN-Transformer Models for Fruit Analysis	9
2.4 Dual-Stage Ensemble Learning Approaches in Fruit Quality Assessment	9
2.5 Summary of Research and Identified Gaps	10
2.6 Tools and Frameworks Used	11
2.7 Consolidated Comparison Table of Prior Research	12
3. SYSTEM ANALYSIS	**14**
3.1 Existing System	14
3.1.1 Disadvantages of Existing System	16
3.2 Proposed System	17
3.3 Feasibility Study	19
4. SYSTEM REQUIREMENTS	**21**
4.1 Software Requirements	21
4.2 Requirement Analysis	21
4.3 Hardware Requirements	22
4.4 Software Description	22
5. SYSTEM DESIGN	**24**
5.1 Design Overview	24
5.2 System Architecture	24
5.3 Functional Modules	27
5.4 UML Diagrams and Data Flow	28
5.5 Design Decisions and Considerations	29
5.6 Model Building	29
5.7 Classification	30
6. METHODOLOGY	**31**
6.1 Dataset	32
6.1.1 Dataset Preparation	33

6.2 Backbone Models 36
6.2.1 ConvNeXt-Tiny (Stage 1 -Fruit Type Classification) 36
6.2.2 EfficientNetV2-B0 (stage 2 -Fruit Quality Grading) 37
6.3 Ensemble Strategy I – Soft Voting 38
6.4 Ensemble Strategy II – Logistic Regression Meta-Ensemble 38
6.5 Inference Workflow 39
6.6 Computational Setup 41

7. IMPLEMENTATION 42
7.1 Environment Setup and Dependencies 42
7.2 Dataset Paths and Class Definitions 43
7.3 Model Initialization 44
7.4 Meta-Ensemble Model Development 45
7.5 Model Evaluation and Visualization 45
7.6 Single Image Inference and Model Saving 46
7.7 Web Integration **47**
7.7.1 System Architecture 47
7.7.2 User Interface Components 48
7.7.3 Backend Workflow (High-level) 48
7.7.4 app.py (Flask application) 49

8. RESULT ANALYSIS 64
8.1 Quantitative Performance Evaluation 64
8.2 Training and Validation Behaviour 67
8.3 Confusion Matrix Analysis 69
8.4 Qualitative Visualization of Preprocessing 71
8.5 Overall Discussion of Results 72
8.6 Functional Test Summary Table 73

9. OUTPUT SCREENS 74
9.1 Home Page 75
9.2 Image Analysis Result Page 76
9.3 Fruit Quality AI – Analysis History Dashboard 77
9.4 Invalid Image Detection (Validation Feature) 78
9.5 Workflow Summary 80

10.CONCLUSION **81**

11.FUTURE SCOPE **83**

12.REFERENCES **85**

LIST OF FIGURES

Figure Number	Description	Page Number
Fig 1.1	Dual-Stage Deep Learning Model for Fruit Quality Prediction	2
Fig 3.1	Workflow of the Dual-Stage Fruit Type and Quality Prediction Model	15
Fig 5.1	System Architecture of the Fruit Quality Detection Model	26
Fig 5.2	Activity Flow Diagram	28
Fig 6.1	Sample images from each category	33
Fig 6.2	Illustration of CutMix Augmentation for Fruit Images	35
Fig 6.3	Illustration of MixUp Augmentation for Fruit Images	35
Fig 6.4	Illustration of AugMix Augmentation for Fruit Images	36
Fig 8.1	Model Accuracy Graph	67
Fig 8.2	Model Loss Graph	67
Fig 8.3	Model Accuracy Per Epoch	68
Fig 8.4	Model Loss Per Epoch	68
Fig 8.5	Normalized Confusion Matrix of Fruit Type Classification	69

Fig 8.6	Normalized Confusion Matrix for Fruit Quality Classification	70
Fig 8.7	Dataset Augmentation for Improved Model Generalization	71
Fig 9.1	Home Page Interface of Application	75
Fig 9.2	Fruit Quality AI – Analysis Results Dashboard	76
Fig 9.3	Fruit Quality AI – Analysis History Dashboard	77
Fig 9.4	Invalid Image Detection and Validation Handling Screen	79

LIST OF TABLES

Section Number	Table Name / Description	Page Number
2.7	Table 2.1 Consolidated Comparison Table of Prior Research	12
5.3	Table 5.1 Functional Modules	27
8.1	Table 8.1 comparative performance across all models	66
8.6	Table 8.2 Functional Test Summary Table	73

1. INTRODUCTION

These days, technology and artificial intelligence have gradually become inseparable parts of our lives. It is hard to imagine a single day without the involvement of smart systems, automation, or digital tools that simplify our work and decision-making. In the field of agriculture, technological advancements have brought remarkable transformation, shifting traditional farming methods toward smart and automated systems. As the demand for high-quality produce continues to grow, fruit classification and quality prediction have become essential aspects of modern agricultural practices. Farmers, industries, and researchers are now using deep learning-based techniques to evaluate fruit types and their quality levels with greater accuracy and speed.

However, this process is not without challenges. Variations in lighting, color, texture, and ripeness make fruit recognition and grading complex tasks. To address these issues, our project introduces a dual-stage deep learning framework that employs ConvNeXt-Tiny for fruit classification and EfficientNetV2-B0 for fruit quality grading. This combination enables the model to perform both identification and evaluation effectively, delivering superior performance in real-world scenarios.

Fruit quality prediction involves analyzing key visual features such as color consistency, surface texture, and shape to determine the quality category — *good*, *average*, or *poor*. Deep learning models, especially convolutional neural networks (CNNs), have shown exceptional ability in learning these fine details automatically from images. In our project, ConvNeXt-Tiny accurately identifies the fruit type, while EfficientNetV2- B0 grades the fruit's condition based on visual indicators. Together, these models create a robust and efficient system capable of supporting real-time fruit sorting and grading — contributing to improved productivity, reduced manual effort, and enhanced precision in modern agriculture.

.

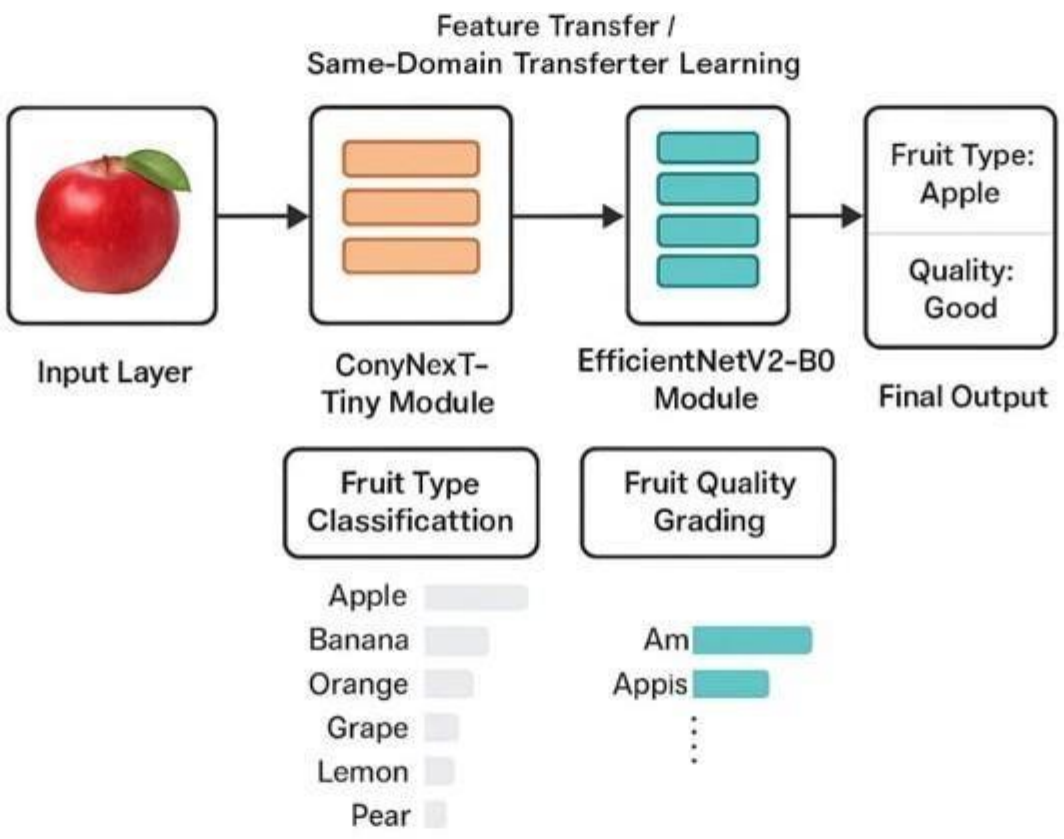


Fig 1.1: Dual-Stage Deep Learning Model for Fruit Quality Prediction

One of the major challenges in India's agricultural and food supply chain is the lack of consistent and accessible fruit quality assessment mechanisms. While large-scale markets, export hubs, and food processing industries often employ trained inspectors and automated sorting systems, small farmers, rural markets, and local vendors frequently rely on manual inspection, which is subjective and error-prone. Variations in lighting, human judgment, fatigue, and lack of expertise can lead to inconsistent quality grading, financial loss, and reduced market value. Additionally, repeated manual inspections and quality disputes increase operational costs and delay decision-making, highlighting the need for affordable and automated fruit quality evaluation systems.

Recent advancements in Deep Learning and Artificial Intelligence have significantly transformed image-based analysis in agriculture by enabling automated, fast, and highly accurate visual inspection systems. Convolutional Neural Networks (CNNs) have demonstrated strong capability in learning hierarchical features such as color, texture, shape, and surface defects from fruit images, making them effective for fruit classification tasks. More recently, transformer-based architectures have gained attention due to their ability to model global contextual relationships and long-range dependencies within images. By combining CNN-based feature extraction with transformer-based contextual modeling, hybrid deep learning systems achieve improved robustness and generalization. Furthermore, ensemble-based learning strategies enhance reliability by integrating predictions from multiple models, thereby reducing misclassification and improving decision stability.

This research proposes a dual-stage deep learning framework that integrates ConvNeXt-Tiny and EfficientNetV2-B0 for automated fruit type classification and quality grading. The system employs comprehensive preprocessing techniques, including resizing, normalization, and advanced data augmentation, to improve input quality and enhance feature learning under varying real-world conditions. The ConvNeXt-Tiny model efficiently captures discriminative visual features related to fruit shape and texture, while EfficientNetV2-B0 focuses on fine-grained quality indicators such as surface uniformity, discoloration, and visible defects. By structuring the framework in a staged manner, the system achieves both high accuracy and computational efficiency.

To support real-world usability, the proposed framework is deployed as a Flask-based web application, enabling users to upload fruit images and receive instant predictions of fruit type and quality category. The system incorporates robust image validation to filter unsupported or low-quality inputs and provides clear, interpretable outputs suitable for farmers, vendors, and quality inspectors. With its ability to handle multiple fruit categories and deliver rapid, reliable assessments, the proposed system has the potential to improve post-harvest quality control, reduce economic loss, and support data-driven decision-making in agricultural practices. As AI adoption continues to expand in agriculture, such intelligent fruit analysis systems are expected to play a vital role in advancing automation, efficiency, and quality assurance across the supply chain.

1.1 Motivation

Agriculture is a vital sector of the global economy, and fruit production holds significant nutritional and commercial importance. Ensuring consistent fruit quality is essential for consumer satisfaction and market competitiveness; however, traditional fruit classification and quality assessment methods rely heavily on manual inspection. Such approaches are time-consuming, labor-intensive, and often inconsistent due to human subjectivity, fatigue, and varying expertise. With recent advances in artificial intelligence and deep learning, automated image-based analysis has emerged as a reliable alternative to manual grading systems. Convolutional Neural Networks (CNNs) have demonstrated strong capability in learning visual features such as color, texture, and shape, making them highly effective for fruit classification and quality prediction. This project is motivated by the need to leverage these advanced techniques to improve accuracy, consistency, and efficiency in agricultural quality assessment.

Another key motivation for this work is the growing demand for high-quality fruits in both domestic and export markets, where strict quality standards must be met at scale. Manual grading methods struggle to maintain uniformity and speed in large production environments, often leading to post-harvest losses and inefficiencies in the supply chain. An intelligent automated system can provide fast, objective, and scalable fruit grading, reducing errors and improving overall productivity. By integrating ConvNeXt-Tiny for fruit type classification and EfficientNetV2-B0 for quality grading, the proposed system achieves reliable performance with reduced computational cost, making it suitable for real-time applications. Furthermore, the inclusion of a simple and user-friendly web interface promotes accessibility for farmers and agricultural industries, bridging the gap between advanced deep learning research and real-world agricultural practices. Overall, this project aims to support smart and sustainable agriculture by minimizing manual effort, reducing wastage, and enhancing quality control through AI-driven automation.

1.2 Problem Statement

Traditional fruit classification and quality assessment methods rely heavily on manual inspection, which is time-consuming, labour-intensive, and highly dependent on human judgment. Factors such as fatigue, lack of consistency, and varying experience levels among inspectors often lead to inaccurate grading. Additionally, environmental variations like lighting conditions and background noise further reduce the reliability of manual evaluation.

With the increasing demand for high-quality fruits in both domestic and international markets, maintaining consistent quality standards has become a major challenge. Manual grading systems are unable to efficiently handle large volumes of fruits in real-time, resulting in delays, increased post-harvest losses, and reduced operational efficiency. These limitations negatively impact farmers, distributors, and the overall agricultural supply chain.

Therefore, there is a critical need for an automated, accurate, and scalable solution that can reliably classify fruit types and predict their quality. The lack of intelligent systems capable of performing real-time fruit analysis highlights the problem addressed by this project, which aims to reduce human dependency, minimize errors, and improve efficiency through deep learning–based fruit classification and quality prediction.

1.3 Objectives

The main objective of this project is to develop an automated and intelligent system for accurate fruit classification and quality prediction using deep learning techniques. The system aims to identify fruit types and evaluate their quality based on visual features such as color, texture, and shape. By integrating ConvNeXt-Tiny and EfficientNetV2-B0 models, the project ensures high accuracy with efficient computation. Overall, the objective is to reduce manual effort, improve grading consistency, and support smart agricultural practices The specific objectives of the project are as follows:

- To preprocess fruit images using resizing, normalization, data augmentation, and enhancement techniques to ensure consistent image quality and improved feature representation for effective model training.

- To fine-tune ConvNeXt-Tiny and EfficientFormerV2-B0 Models independently, leveraging their complementary strengths—ConvNeXt-Tiny for efficient fruit type classification and EfficientFormerV2-B0 for accurate fruit quality grading
- To design and implement dual-stage learning framework that sequentially performs fruit type identification followed by quality Assessment, ensuring reliable and Structured prediction outcome .
- To evaluate the performance of the proposed system using comprehensive metrics such as accuracy, precision, recall, F1-score, confusion matrix, and ROC–AUC, ensuring balanced assessment across different fruit categories and quality levels..
- To deploy the finalized model through a Flask-based web application that allows users to upload fruit images and receive real-time classification and quality Grading results in a accessible and user-friendly manner..
- To assist farmers ,vendors, and agricultural industries by reducing manual inspection effort, minimizing, grading errors, and enabling fast, objective fruit quality assessment, thereby supporting efficient sorting, reduced wastage, and improved agricultural productivity..

2. LITERATURE SURVEY

2.1 Deep Learning in Agricultural Image Analysis

Deep learning has significantly transformed agricultural image analysis by enabling automated and accurate interpretation of visual data related to crops and fruits. Unlike traditional image processing techniques that rely on handcrafted features, deep neural networks learn discriminative representations directly from raw images, making them highly effective for large-scale agricultural applications.

Gupta et al. [1] reviewed multiple deep learning approaches for fruit classification and highlighted that Convolutional Neural Networks (CNNs) consistently outperform conventional machine vision methods. Their study emphasized that transfer learning using pre-trained architectures such as VGG16, ResNet50, and InceptionV3 improves classification accuracy, especially when agricultural datasets are limited in size.

Similarly, Das and Basu [4] introduced the FruitNet dataset, a large-scale, well-annotated benchmark for multi-class fruit recognition. Their work demonstrated that CNN-based architectures achieve superior performance compared to shallow machine learning models when trained on diverse and large datasets. The authors also noted that attention mechanisms help focus on defective or visually critical fruit regions, which is essential for quality grading.

Prasad and Reddy [10] provided a comprehensive overview of deep learning advancements in agriculture, highlighting how lightweight CNNs and transformer-based models enable high accuracy with reduced computational cost. Their study emphasized the growing role of AI-driven systems in automating fruit sorting, grading, and pricing in modern agricultural markets.

2.2 CNN-Based Approaches for Fruit Classification and Quality Grading

CNN-based architectures have become the foundation of automated fruit classification and grading systems due to their strong capability to learn hierarchical features such as color, texture, shape, and surface defects.

Kumar et al. [2] proposed a fruit quality detection system using deep CNNs combined with image enhancement techniques. Their work addressed dataset imbalance issues and demonstrated that preprocessing methods such as histogram equalization and Gaussian filtering significantly improve visual clarity and classification accuracy.

Patel and Mehta [3] explored multi-fruit classification using pre-trained CNN models including EfficientNet and MobileNetV3. Their comparative analysis using accuracy, precision, recall, and F1-score showed that transfer learning enables efficient performance even with limited computational resources.

Roy and Sinha [5] implemented ResNet101 and EfficientNetB0 to predict fruit ripeness and quality levels. Their multi-stage CNN pipeline successfully captured minor variations in color, texture, and size, achieving an overall accuracy of 97%. The study highlighted the importance of same-domain transfer learning for improving robustness in fruit grading tasks.

To Despite these successes, CNN-based approaches face challenges such as sensitivity to lighting conditions, background noise, and difficulty in capturing global contextual relationships. These limitations motivate the exploration of hybrid and transformer-based architectures.

2.3 Hybrid CNN–Transformer Models for Fruit Analysis

Recent studies have shown that combining CNNs with transformer-based architectures improves performance by integrating local feature extraction with global contextual modeling.

Tanwar and Joshi [6] conducted a comparative study of ConvNeXt-Tiny, EfficientNetV2, and Vision Transformer (ViT) for agricultural image classification. Their results showed that ConvNeXt-Tiny offers an effective balance between speed and accuracy, while EfficientNetV2 excels at identifying subtle visual differences between healthy and defective fruits.

Verma et al. [7] proposed a hybrid CNN–Transformer network for fruit defect detection. Their architecture effectively captured both spatial and global dependencies, achieving high accuracy in detecting defects caused by bruising and fungal infections. The study confirmed that hybrid models outperform standalone CNN or transformer architectures in complex fruit grading scenarios.

2.4 Dual-Stage and Ensemble Learning Approaches in Fruit Quality Assessment

Ensemble and multi-stage learning frameworks have gained popularity in agricultural applications due to their ability to improve robustness, reduce misclassification, and enhance generalization.

Khan et al. [8] proposed an end-to-end fruit grading system using deep CNNs deployed on edge devices. Their system achieved near real-time performance in grading fruits into multiple quality categories, emphasizing the importance of computational efficiency for deployment in farms and warehouses.

The dual-stage framework proposed by the authors in [9] introduced a two-phase learning strategy—fruit type classification followed by quality grading. The system utilized EfficientNetV2-B0 for robust feature extraction and ConvNeXt-Tiny for efficient prediction, achieving superior generalization and reduced false grading rates compared to single-model approaches.

Motivated by these works, the present project adopts a dual-stage deep learning framework that integrates ConvNeXt-Tiny for fruit type classification and EfficientNetV2-B0 for fruit quality grading. This approach leverages transfer learning and advanced augmentation techniques to achieve reliable, balanced, and computationally efficient performance suitable for real-world agricultural deployment.

2.5 Summary of Research and Identified Gaps

Existing research demonstrates that deep learning models, particularly CNN-based and hybrid architectures, have significantly improved automated fruit classification and quality grading. CNNs are effective in capturing local visual features such as color, texture, shape, and surface defects, while recent hybrid and transformer-inspired models enhance the understanding of broader contextual and structural patterns. Ensemble and multi-stage learning approaches further improve robustness by combining multiple models, thereby reducing misclassification and improving overall prediction reliability.

Despite significant progress, several research gaps remain in automated fruit analysis. CNN-based models often struggle with visually similar fruits and subtle quality differences due to limited global context awareness. Transformer-based architectures require large annotated datasets and high computational resources, restricting real-time and edge deployment. Moreover, many existing fruit grading systems rely on single models or simple fusion techniques, which fail to fully utilize the complementary strengths of different architectures.

Another major challenge is class imbalance in fruit datasets, especially among quality categories where defective samples are limited. This imbalance leads to biased predictions and reduced accuracy for minority quality classes. Additionally, there is limited research on dual-stage or transfer learning–based frameworks that separately address fruit classification and quality grading. Maintaining both computational efficiency and strong generalization remains an open challenge.

To address these gaps, this project proposes a dual-stage deep learning framework integrating ConvNeXt-Tiny for fruit classification and EfficientNetV2-B0 for quality grading.

2.6 Tools and Frameworks Used

The implementation of the proposed Fruit Classification and Quality Grading system utilized a modern and efficient technology stack to support model development, training, evaluation, and deployment.

- **Python** – Used as the primary programming language for implementing deep learning models and system logic.
- **TensorFlow & KerasCV** – Employed for building, training, and fine-tuning the ConvNeXt-Tiny and EfficientNetV2-B0 models using transfer learning.
- **NumPy & Pandas** – Used for data manipulation, preprocessing, label handling, and dataset management.
- **OpenCV & PIL** – Applied for image loading, resizing, normalization, and preprocessing operations.
- **Albumentations / KerasCV Augmentations** – Utilized to implement advanced data augmentation techniques such as CutMix, MixUp, and AugMix.
- **Scikit-learn** – Used for performance evaluation metrics including accuracy, precision, recall, F1-score, confusion matrix, and ROC–AUC computation.
- **Matplotlib & Seaborn** – Used for visualizing training and validation curves, confusion matrices, and performance results.
- **Google Colab (CPU/GPU Environment)** – Used for model training and experimentation, providing scalable computational resources.

This integrated technology stack enabled efficient experimentation, robust model evaluation, and seamless deployment of the AI-based fruit classification and quality grading system in a user-friendly web application.

2.7 Consolidated Comparison Table of Prior Research

Table 2.1: Consolidated Comparison Table of Prior Research

Study / Approach	Model Type	Dataset	Strengths	Limitations	Reference
CNN-based Fruit Classifier	CNN	Fruit image datasets	Good feature extraction; improved over traditional ML	Requires large dataset; overfitting on small data	[1]
Deep CNN for Fruit Quality Detection	CNN	Agricultural datasets	Effective quality grading (good vs bad)	Limited generalization to different lighting	[2]
Multi-Fruit Classification using Transfer Learning	EfficientNet, MobileNet	Multi-fruit datasets	High accuracy with limited data	Computational cost on larger models	[3]
FruitNet Framework	CNN	FruitNet dataset	Large-scale dataset improves robustness	Imbalanced quality classes	[4]
Ripeness Detection using ResNet & EfficientNet	CNN	Fruit ripeness datasets	High accuracy (97% reported)	Needs fine-tuning for each fruit type	[5]
ConvNeXt-Tiny based Agricultural Study	CNN (ConvNeXt-Tiny)	Crop & fruit datasets	Balanced speed and accuracy	Requires GPU for training	[6]

Hybrid CNN–Transformer Model	CNN + Transformer	Fruit defect datasets	Captures local & global features	Higher computational complexity	[7]
Edge-based Fruit Grading System	Deep CNN	Real-time fruit images	Near real-time performance	Limited scalability	[8]
Dual-Stage Fruit Classification Framework	EfficientNetV2 + ConvNeXt	Fruit quality datasets	Separates type & quality → better accuracy	Slightly complex pipeline	[9]

3. SYSTEM ANALYSIS

3.1 Existing System

Traditional fruit classification and quality grading systems have primarily relied on manual inspection performed by human graders in farms, warehouses, and marketplaces. Although experienced workers can visually assess fruit type and quality based on color, size, texture, and surface defects, this manual process is::

- Time-consuming,
- Subjective, and inconsistent,
- Prone to human fatigue and error.

Manual grading becomes especially challenging when fruits exhibit subtle quality variations, such as early-stage bruising, minor discoloration, fungal spots, or ripeness differences. These factors often lead to inaccurate grading, delayed decision-making, increased post-harvest losses, and reduced market value.

To overcome these limitations, earlier automated systems adopted traditional machine learning techniques, including::

- Support Vector Machines (SVM)
- Random Forests (RF)
- Decision Trees
- k-Nearest Neighbors (k-NN)

These systems relied heavily on handcrafted feature extraction, such as:Color histograms

- Texture descriptors (GLCM, LBP)
- Shape and size features
- Geometric and edge-based attributes
- Color histograms

While such methods achieved moderate success under controlled conditions, they suffered from several drawbacks:

- Manual feature extraction required expert knowledge
- Poor generalization to new datasets and lighting conditions
- Sensitivity to background noise and image quality

The advancements in Deep Learning, Convolutional Neural Networks (CNNs) significantly improved fruit image analysis by automatically learning discriminative features directly from images. Models such as VGG, ResNet, Inception, and DenseNet demonstrated higher accuracy in fruit recognition and grading compared to traditional ML approaches.

However, CNN-based systems also introduced challenges:

- Requirement of large labeled datasets
- Overfitting on small or imbalanced fruit datasets
- Limited ability to capture global contextual relationships
- High computational and hardware requirements

To address data scarcity, transfer learning became widely used, where pretrained models were fine-tuned on fruit image datasets. This approach improved convergence speed and classification accuracy compared to training from scratch. Nevertheless, standalone CNN architectures continued to struggle in multi-fruit classification and quality grading, especially when fruits shared similar visual characteristics.

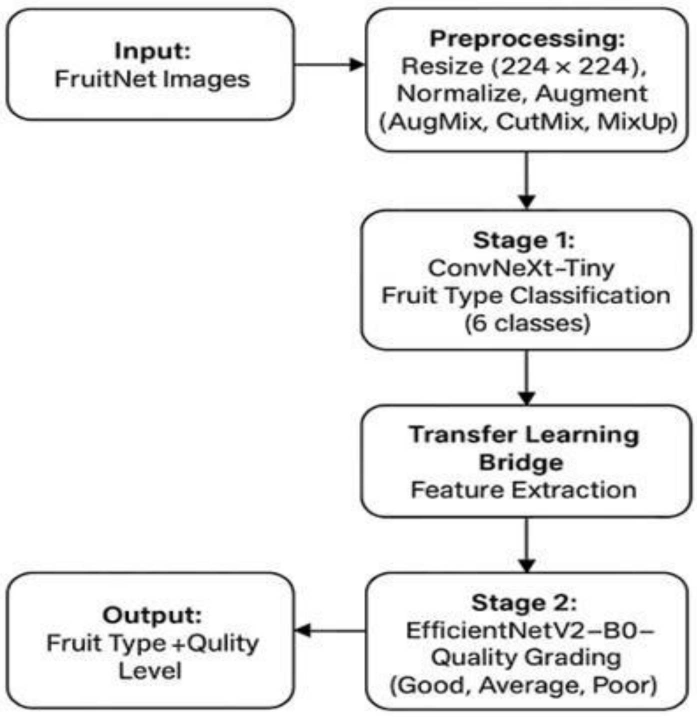


Fig 3.1: Workflow of the Dual-Stage Fruit Type and Quality Prediction Model

The traditional Workflow The conventional automated fruit grading pipeline generally consists of the following stages:

1. **Preprocessing**
   Improves image quality through resizing, normalization, noise removal, and illumination correction.
2. **Segmentation**
   Separates the fruit region from the background for focused analysis.
3. **Feature Extraction**
   Extracts important characteristics such as color distribution, texture patterns, shape, and surface defects.
4. **Classification**
   Assigns fruit type and quality category using ML or CNN-based models.

Despite progress, these systems remain sensitive to lighting variations, background clutter, occlusions, reflections, and surface noise, limiting their real-world reliability.

3.1.1 DISADVANTAGES OF EXISTING SYSTEM

Despite technological improvements, existing automated fruit classification and quality grading systems suffer from several critical limitations:

- **Dependence on Large Annotated Datasets**
  Deep learning models require large labeled datasets, but collecting and annotating agricultural images is time-consuming and costly.
- **High Computational Requirements**
  Advanced CNN architectures demand powerful GPUs, limiting deployment in farms and small-scale agricultural facilities.
- **Overfitting on Small or Imbalanced Datasets**
  Quality datasets often contain fewer defective samples, leading to:
  Biased learning
  Reduced accuracy for minority quality
  Poor generalization

- **Limited Global Context Understanding**
  CNNs focus on local features and struggle to capture global fruit structure, affecting quality grading accuracy.
- **Sensitivity to Environmental Artifacts**
  Fruit images often contain:
  - Shadows
  - Background clutter
  - Reflections
  - Blur
  - Uneven illumination

  These degrade model performance and reduce reliability.
- **Poor Adaptability Across Imaging Devices**
  Variations in cameras, resolution, and lighting lead to inconsistent predictions.
- **Suboptimal Segmentation Accuracy**

  Traditional segmentation methods struggle with irregular fruit shapes and complex backgrounds, negatively affecting classification.
  These limitations highlight the need for a **robust, scalable, and efficient system** suitable for real-world agricultural environments.

3.2 Proposed System

The proposed system introduces a dual-stage deep learning framework for fruit type classification and quality grading, as shown in Figure 3.2. The framework integrates two complementary architectures:

- ConvNeXt-Tiny for efficient and accurate fruit type classification
- EfficientNetV2-B0 for fine-grained fruit quality grading

This separation allows each model to specialize in its respective task, improving overall performance and stability.

The processing pipeline begins with preprocessing, where fruit images undergo resizing, normalization, and advanced data augmentation techniques such as CutMix, MixUp, and AugMix to address dataset imbalance and enhance generalization.

In Stage 1, ConvNeXt-Tiny extracts discriminative features related to fruit shape, color, and texture to accurately identify fruit type. The learned features are then transferred through a transfer learning bridge to support Stage 2, where EfficientNetV2-B0 performs quality grading by analyzing surface defects, discoloration, and texture irregularities.

This dual-stage architecture improves classification robustness, reduces error propagation, and enables efficient inference suitable for real-time applications.

Advantages of the Proposed System

- **High Accuracy and Robustness**
  Dual-stage learning improves prediction stability.
- **Better Generalization**
  Advanced augmentation and transfer learning enhance performance under varied conditions.
- **Improved Quality Assessment**
  EfficientNetV2-B0 captures subtle surface defects and quality indicators.
- **Reduced Overfitting**
  Stage-wise learning and augmentation mitigate data imbalance effects.
- **Scalable and Extensible**
  The system can be extended to additional fruit types and quality categories.
- **Deployment Ready**
  Integrated into a Flask-based web application for real-time analysis.

Overall, the proposed dual-stage framework effectively addresses the limitations of traditional fruit grading systems by combining efficient feature extraction, advanced data augmentation, and transfer learning. By separating fruit type classification and quality grading into dedicated stages, the system achieves higher accuracy, improved robustness, and better generalization under real-world agricultural conditions. Its lightweight architecture and deployment-ready design make it suitable for practical applications such as automated fruit sorting, quality control, and decision support in modern agricultural supply chains

3.3 Feasibility Study

The feasibility of the proposed system is analyzed across **Technical, Operational, and Economic** aspects.

1. Technical Feasibility

- **Automated Feature Extraction:**
  ConvNeXt-Tiny and EfficientNetV2-B0 automatically learn discriminative features related to fruit type and quality, eliminating the need for manual feature engineering.
- **Accurate and Robust Classification:**
  The dual-stage framework improves classification reliability by separating fruit type identification and quality grading.
- **Reduced Overfitting:**
  Advanced data augmentation techniques such as CutMix, MixUp, and AugMix improve model generalization on imbalanced datasets.
- **Scalable Architecture:**
  The system can be extended to additional fruit varieties and quality levels without significant architectural changes.
- **Efficient Transfer Learning:**
  Pretrained model weights reduce training time and enable high accuracy even with limited agricultural data.

2. Operational Feasibility

- **Ease of Integration:**
  A Flask-based web interface allows users to upload fruit images and receive predictions instantly.
- **High User Acceptance:**
  Clear display of fruit type, quality level, and confidence scores enhances user trust and usability.
- **Low Maintenance Requirements:**
  Model updates or retraining can be performed efficiently without major system modifications.

- **Support for Non-Expert Users:**

  The system provides reliable quality assessment without requiring technical or agricultural expertise.

3. **Economic Feasibility**

- **Cost-Effective Development:**

  Transfer learning reduces data collection and computational costs.

- **Optimized GPU Usage:**

  Lightweight architectures such as ConvNeXt-Tiny and EfficientNetV2-B0 enable efficient inference with reduced hardware requirements.

4. SYSTEM REQUIREMENTS

4.1 Software Requirements

1. **Operating System** : Windows 11 (64-bit)
2. **Hardware Accelerator** : CPU (GPU optional but recommended for training)
3. **Programming Language** : Python
4. **Python Environment / Tools** : Google Colab Pro, Flask
5. **Browser** : Any latest browser (e.g., Google Chrome, Firefox, Edge)

4.2 Requirement Analysis

The proposed Fruit Classification and Quality Grading System is designed to develop an efficient, accurate, and automated deep learning-based solution capable of identifying multiple fruit types and grading their quality levels using image inputs. The system integrates ConvNeXt-Tiny for fruit type classification and EfficientNetV2-B0 for fruit quality grading in a dual-stage architecture to improve robustness, accuracy, and generalization.

The application allows users to upload fruit images through a web interface, where the system first performs input validation to ensure that only supported fruit images are processed. The uploaded images undergo preprocessing steps such as resizing, normalization, and advanced data augmentation (CutMix, MixUp, and AugMix) to enhance feature learning and improve model performance. The system then performs inference in two stages: first predicting the fruit type and then determining the corresponding quality category. The final output displays the fruit name, quality level (e.g., Good, Average, Poor), and confidence scores.

The backend is implemented in Python using the Flask framework, which handles image preprocessing, model inference, and communication between frontend and backend components. The frontend is developed using HTML, CSS, Bootstrap, and JavaScript, ensuring a clean, responsive, and user-friendly interface suitable for agricultural and industrial users.

Non-functional requirements emphasize system performance, usability, scalability, and reliability. The application must generate predictions within seconds and handle invalid or unsupported images gracefully by providing clear error messages. The system should operate efficiently under different lighting conditions and background variations.

Model development and training require frameworks such as TensorFlow/Keras, scikit-learn, NumPy, Pandas, and OpenCV, along with sufficient computational resources for processing fruit image datasets. A properly labeled fruit dataset containing both type and quality annotations is essential for training, validation, and testing. The system can be deployed locally or on a cloud platform, allowing access through any standard web browser. To ensure practical adoption, the interface is designed to be intuitive and accessible even for users with minimal technical expertise.

4.3 Hardware Requirements

1. **System Type** : 64-bit Operating System, x64-based processor
2. **Cache Memory** : 4 MB
3. **RAM** : 16 GB
4. **Hard Disk** : 8 GB free space
5. **GPU** : Intel® Iris® Xe Graphics (sufficient for inference; training recommended on cloud GPU)

4.4 Software Description

The Fruit Classification and Quality Grading System is developed using a combination of modern software tools and frameworks to ensure high accuracy, efficiency, and scalability.

The recommended platform is Windows 10/11 (64-bit Operating System), ensuring compatibility with development tools and stable execution of deep learning workflows. Model inference can be performed on a CPU, while training is conducted using Google Colab, which provides access to GPU resources for faster computation.

The core development is carried out using Python, selected for its extensive ecosystem of machine learning and deep learning libraries. Google Colab is used for model training, experimentation, and evaluation. The trained models ConvNeXt-Tiny for fruit classification and EfficientNetV2-B0 for fruit quality grading are integrated into a Flask-based backend, which manages image uploads, preprocessing, inference, and communication with the user interface.

The frontend is developed using:

- HTML5, CSS3, and Bootstrap for an intuitive and responsive interface
- JavaScript for interactive elements and smooth user experience

The application runs smoothly on any modern web browser such as Google Chrome, Mozilla Firefox, or Microsoft Edge.

For machine learning and deep learning tasks:

- **TensorFlow/Keras** is used to is used to build and train the ConvNeXt-Tiny and EfficientNetV2-B0 models
- **scikit-learn** is used for performance evaluation metrics.
- **OpenCV** performs or image preprocessing tasks such as resizing, normalization, and validation
- **NumPy** support efficient numerical computations and dataset handling

Visualization tools such as **Matplotlib** and **Seaborn** are used to plot accuracy, loss curves, and confusion matrices during model evaluation.

Together, these tools enable a robust, scalable, and , and efficient system for automated fruit classification and quality grading, suitable for real-world agricultural deployment.

5. SYSTEM DESIGN

5.1 Design Overview

The proposed Fruit Classification and Quality Grading System is implemented as a Flask-based web application that integrates deep learning models with an intuitive and user-friendly interface. The system adopts a dual-stage architecture that combines ConvNeXt-Tiny for fruit type classification and EfficientNetV2-B0 for fruit quality grading, enabling accurate and reliable predictions from uploaded fruit images.

The system follows a structured **client–server architecture**. The frontend allows users to upload fruit images, initiate analysis, and view classification and quality grading results along with confidence scores. The backend is responsible for handling image validation, preprocessing (resizing, normalization, augmentation where applicable), model inference, and rendering the final predictions to the user interface.

This modular design ensures scalability, maintainability, and efficient performance. By separating the user interface from the deep learning inference pipeline, the system enables smooth interaction while keeping the computational processes optimized and secure. The architecture supports real-time fruit analysis and is designed for easy deployment in agricultural and industrial environments.

5.2 System Architecture

The overall architecture of the Fruit Quality AI application is organized into five major layers, each responsible for a specific function in the end-to-end fruit classification and quality grading pipeline.

1. User Interface Layer

This layer provides a clean, responsive, and user-friendly interface that allows users to interact with the system easily.

- Upload fruit images
- Trigger automated analysis
- View previous results

Templates (HTML/CSS) rendered with Flask's Jinja engine support pages such as: index.html, upload.html, result.html.

2. Flask Application Layer (Backend Controller)

Acts as the middle layer between the user interface and the deep learning engine.
It manages:

- Routing (/, /home, /samples, /history)
- File upload handling
- Input validation
- Preprocessing pipeline invocation
- Model inference calls
- Rendering prediction results
- Storing analysis history

3. Preprocessing and Model Layer

Before inference, uploaded fruit images preprocessing to ensure consistent model input:

- Image resizing to 224×224
- Normalization
- Format validation (JPG/PNG)
- Quality checks (valid fruit vs unsupported image)

Advanced augmentation techniques such as CutMix, MixUp, and AugMix are applied during training to improve model generalization and robustness.

4. Deep Learning Inference Layer

This is the core intelligence layer of the system.
The architecture follows a **dual-stage pipeline**:

1. **Stage 1 – ConvNeXt-Tiny**
    - Performs fruit type classification
    - Identifies fruit categories such as Apple, Banana, Guava, Lemon, Lime, Orange, and Pomegranate .

2. **Stage 2 – EfficientNetV2-B0**
   - Performs fruit quality grading
   - Classifies quality levels such as Good, Bad, or Average

If the uploaded image is invalid or unsupported, the system triggers an Analysis Incomplete alert.

The architectural flow of the complete system is illustrated in **Fig. 5.1**, which shows how an uploaded image travels through the UI → Flask backend → preprocessing → dual-model inference → final result display → History update.

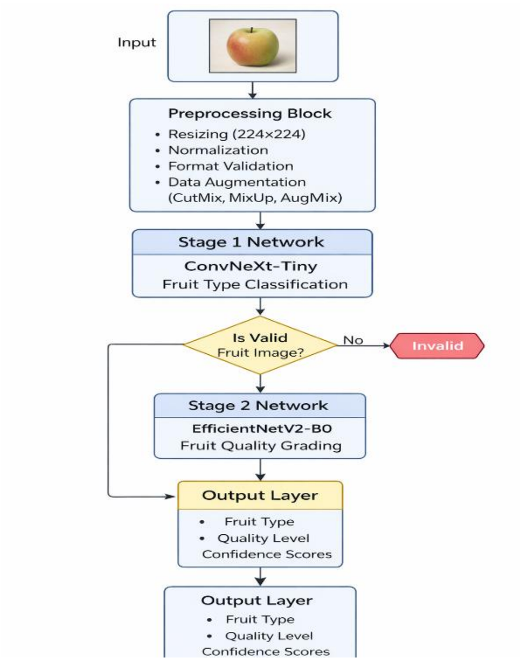


Fig. 5.1: System Architecture of the Fruit Quality Detection Model

The above diagram illustrates the complete workflow of the proposed Fruit Quality AI system. The process begins with the input fruit image uploaded by the user through the web interface.

In the **Preprocessing Block**, the image undergoes resizing to 224×224 pixels to match the model input dimensions. Normalization is applied to stabilize training distribution, followed by format validation to ensure the image is valid. Advanced data augmentation techniques such as **CutMix, MixUp, and AugMix** are used during training to improve robustness and generalization.

This dual-stage architecture ensures accurate fruit identification, reliable quality grading, and efficient real-time deployment for practical agricultural applications.

5.3 Functional Modules

Table 5.1: Functional Modules

Module	Description
User Interface Module	Provides web-based interaction using Flask templates (HTML/CSS/Bootstrap) for uploading fruit images and viewing results.
Image Upload & Validation	Accepts JPG/PNG images (≤6MB), validates file format and size, assigns secure filenames, and stores images in the /static/uploads directory.
Preprocessing Module	Resizes images to 224×224, converts to RGB, normalizes pixel values, and applies model-specific preprocessing (ConvNeXt & EfficientNetV2).
Fruit Type Classification Module	Uses ConvNeXt-Tiny model to classify the uploaded fruit image into predefined fruit categories.
Fruit Quality Detection Module	Uses EfficientNetV2-B0 model to determine fruit quality (Good / Bad / Mixed).
Confidence Threshold Module	Applies predefined confidence thresholds to detect invalid or low-confidence predictions.
Result Storage Module	Stores prediction results (fruit type, quality, confidence scores) in a JSON file (analysis_results.json).
History Management Module	Retrieves and displays previously analyzed fruit images and results through the /history route.

5.4 UML Diagrams and Data Flow

Use Case Overview

Actors:

- User
- System

Use Cases:

- Upload Fruit Image
- Analyze Image
- View Result
- View History
- Delete Record

Sequence of Operations

1. User opens the Fruit Quality AI web application.
2. User uploads one or multiple fruit images.
3. Flask backend validates the file format and image type.
4. The image undergoes preprocessing (resizing, normalization, augmentation).
5. ConvNeXt-Tiny predicts the fruit type.
6. If the image is valid, EfficientNetV2-B0 predicts the fruit quality.
7. The prediction results (type, quality, confidence scores) are stored in the History.
8. The final output is displayed on the user interface.

Activity Flow Diagram

Fig. 5.2 depicts the complete activity flow:

User → Image Upload → Validation → Preprocessing → Stage 1:Fruit Type Classification → Validity Check → Stage 2: Quality Grading→ Store in Database → Display Output

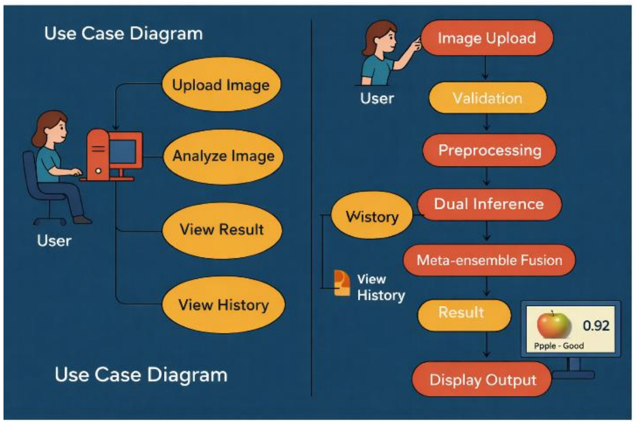


Fig:5.2: Activity Flow Diagram

5.5 Design Decisions and Considerations

- **Flask Framework:**
  Chosen for lightweight backend development and smooth integration with deep learning models.
- **Model Selection:**
  ConvNeXt-Tiny ensures accurate fruit type classification with efficient computation, while EfficientNetV2-B0 provides precise quality grading with strong feature extraction capability.
- **Dual-Stage Architecture:**
  Separating fruit type classification and quality grading improves robustness and reduces misclassification between visually similar fruits.
- **Security Measures:**
  Includes file type validation, format checking, and secure filename handling to prevent invalid inputs.
- **Scalability:**
  New fruit types or quality categories can be added without modifying the overall system structure.

5.6 Model Building

The proposed dual-stage deep learning model combines:

- **ConvNeXt-Tiny**
- **EfficientNetV2-B0**

Each model performs a specific task in the pipeline.

ConvNeXt-Tiny (Stage 1 – Fruit Type Classification)

- Lightweight and computationally efficient
- Extracts color, texture, and shape features
- Classifies fruit into supported categories (Apple, Banana, Guava, Lemon, Lime, Orange, Pomegranate)

EfficientNetV2-B0 (Stage 2 – Fruit Quality Grading)

- Optimized CNN architecture
- Captures fine-grained surface defects

- Classifies quality levels (Good, Bad, Mixed)

Advanced augmentation techniques (CutMix, MixUp, AugMix) are applied during training to improve generalization and handle class imbalance.

5.7 Classification

The final classification stage produces:

- Fruit Type
- Quality Level
- Confidence Scores

The classification process includes:

- SoftMax probability distribution
- Confidence score calculation
- Validity check for non-fruit images

This dual-stage strategy improves overall reliability, reduces confusion between similar fruits, and enhances grading accuracy.

Compared to traditional single-model systems, the proposed framework provides:

- Better generalization
- Higher accuracy
- Improved robustness
- Practical deployment capability for real-world agricultural applications

6. METHODOLOGY

The methodology of this project is designed around a dual-stage deep learning framework for accurate fruit type classification and quality grading. Since many fruits share similar visual characteristics such as color, texture, and shape—and quality differences can be subtle (e.g., minor bruises or surface defects)—the system adopts a structured pipeline that separates fruit identification from quality assessment to improve overall reliability.

To achieve this, the framework integrates ConvNeXt-Tiny for fruit type classification and EfficientNetV2-B0 for fruit quality grading. ConvNeXt-Tiny is selected for its strong feature extraction capability and balanced computational efficiency, enabling accurate recognition of fruit categories such as Apple, Banana, Guava, Lemon, Lime, Orange, and Pomegranate. Once the fruit type is identified and validated, EfficientNetV2-B0 performs fine-grained quality grading (Good, Bad, or Mixed Quality). This staged architecture allows the system to first understand *what fruit it is* and then determine *its quality level,* reducing misclassification and improving grading precision.

The methodology incorporates advanced data augmentation techniques including CutMix, MixUp, and AugMix, along with resizing and normalization. These techniques enhance dataset diversity, reduce overfitting, and improve generalization under varying lighting, backgrounds, and fruit orientations commonly observed in real-world agricultural environments.

Dataset preparation and augmentation, ensuring image uniformity and diversity across different fruit types and quality categories through resizing (224×224), normalization, and advanced augmentation techniques such as CutMix, MixUp, and AugMix.

1. **Independent training** of **ConvNeXt-Tiny** and **EfficientNetV2-B0** models, where ConvNeXt-Tiny is trained for fruit type classification and EfficientNetV2-B0 is trained for fruit quality grading using transfer learning.
2. **Design** of a **dual-stage framework**, where Stage 1 validates and classifies the fruit type before forwarding the image to Stage 2 for detailed quality assessment.

3. **Inference pipeline development**, enabling fast and reliable real-time predictions through a Flask-based web application interface.

This integrated dual-stage framework provides a strong foundation for building an AI-assisted fruit classification and quality grading system, improving accuracy, reducing manual inspection errors, and supporting smart agricultural automation.

6.1 Dataset

The dataset used in this project is the FruitNet – Indian Fruits Dataset with Quality, a publicly available image dataset designed for fruit classification and quality assessment tasks. This dataset contains images of multiple Indian fruit varieties along with their corresponding quality labels, making it suitable for developing a dual-stage fruit type classification and quality grading system.

The dataset is publicly available on Kaggle and can be accessed
https://www.kaggle.com/datasets/shashwatwork/fruitnet-indian-fruits-dataset-with-quality

The FruitNet dataset includes images of commonly consumed fruits such as Apple, Banana, Guava, Lemon, Lime, Orange, and Pomegranate, categorized into different quality levels (e.g., Good, Bad, and Mixed quality). The images are captured under varying lighting conditions, backgrounds, and orientations, reflecting real-world agricultural environments.

All images are provided in standard formats such as JPEG/PNG, ensuring compatibility with deep learning frameworks like PyTorch and TensorFlow. The dataset exhibits natural variability in fruit color, texture, shape, surface defects, bruises, and ripeness levels. This diversity introduces realistic grading challenges and helps improve the generalization capability of the proposed models.

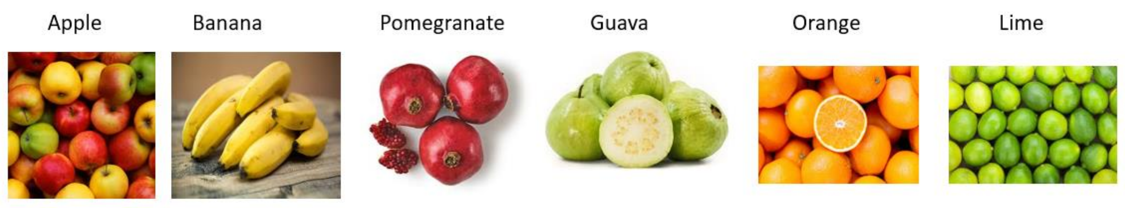


Fig.6.1: Sample images from each category

Before model training and inference, all fruit images undergo preprocessing steps such as resizing (224×224), normalization, format validation, and data augmentation techniques including CutMix, MixUp, and AugMix. These steps ensure uniform input representation and reduce variability caused by differences in lighting conditions, camera angles, background noise, and image acquisition environments.

The FruitNet – Indian Fruits Dataset includes thousands of high-quality fruit images collected under diverse real-world conditions. The dataset contains multiple fruit categories such as Apple, Banana, Guava, Lime, Orange, and Pomegranate, along with different quality levels (e.g., Good, Bad, or Defective).

All images are stored in JPEG/PNG format, ensuring compatibility with deep learning frameworks such as TensorFlow and PyTorch. The dataset contains variations in fruit size, shape, orientation, surface texture, bruises, and background settings, which simulate realistic agricultural and market scenarios. This diversity helps the model generalize effectively to unseen fruit samples in real-world environments.

Before feeding the images into the deep learning models, preprocessing techniques—including resizing, normalization, augmentation, and image enhancement—ensure consistency in input dimensions and improve feature clarity. These steps reduce the impact of illumination differences, shadows, and camera variations, thereby enhancing overall classification and quality grading performance.

6.1.1 Dataset Preparation

Before training and evaluation, the fruit images from the FruitNet – Indian Fruits Dataset undergo a series of preprocessing and augmentation steps to ensure consistent input representation, improve feature learning, and reduce variability caused by different camera angles, lighting conditions, and backgrounds commonly found in real-world agricultural environments.

Preprocessing Steps

The following preprocessing operations are applied uniformly to fruit images:

1. **Resizing to 224 × 224 pixels**
   All images are resized to a fixed resolution of 224 × 224 pixels to ensure compatibility with both ConvNeXt-Tiny (Stage 1: Fruit Type Classification) and EfficientNetV2-B0 (Stage 2: Quality Grading**)** architectures, which require fixed-size inputs.
2. **Pixel Normalization**
   Pixel intensity values are normalized using predefined mean and standard deviation values. This stabilizes the training process and reduces the effect of illumination differences, shadows, and color variations across fruit images.
3. **Format Validation and Tensor Conversion**
   Uploaded images are first validated to ensure they are in supported formats (JPG/PNG). The images are then converted into tensor format to enable efficient GPU-accelerated computation during model training and inference.

Augmentation Technique

To improve model generalization and reduce overfitting, several advanced data augmentation techniques are applied during training. These techniques artificially increase dataset diversity by creating modified variations of existing fruit images, helping the model perform better under real-world agricultural conditions such as varying lighting, angles, and backgrounds.

1. CutMix combines two fruit images by cutting a region from one image and pasting it onto another. The labels are also mixed proportionally.

- Helps the model focus on multiple regions of interest.
- Improves robustness against partial occlusion or damaged fruit areas.

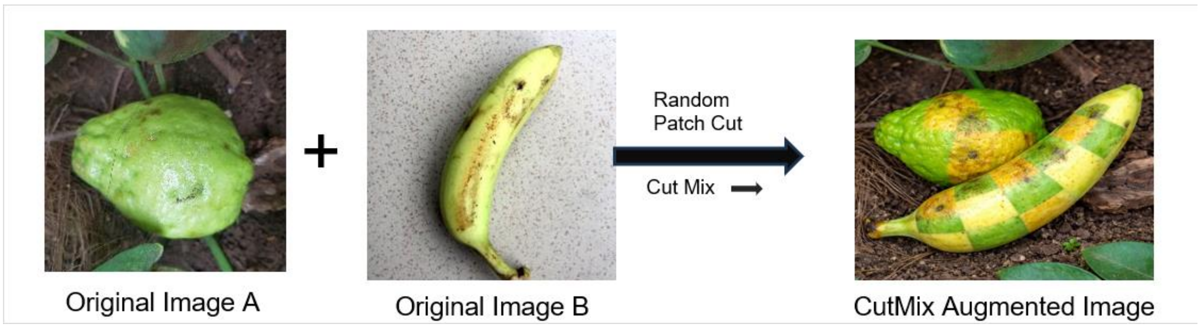


Fig 6.2: Illustration of CutMix Augmentation for Fruit Images

The figure demonstrates the CutMix data augmentation technique, where two original fruit images (Image A and Image B) are combined by cutting a random patch from one image and pasting it onto the other. The resulting CutMix augmented image contains mixed visual features from both fruits, improving model robustness, reducing overfitting, and enhancing generalization for fruit classification and quality grading tasks.

2. **MixUp** creates a new training sample by blending two fruit images and their corresponding labels using a weighted average.
   - Encourages smoother decision boundaries.
   - Improves generalization performance.
   - Reduces model sensitivity to noise and minor variations.

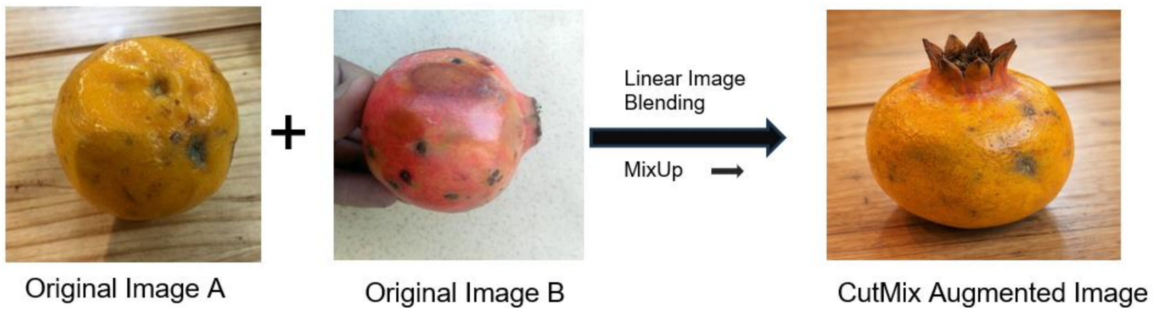


Fig 6.3: Illustration of MixUp Augmentation for Fruit Images

The figure demonstrates the MixUp data augmentation technique, where two original fruit images (Image A and Image B) are linearly blended to create a new synthetic training image. Unlike CutMix, MixUp combines pixel values proportionally, resulting in smoother feature transitions. This technique improves model generalization, reduces overfitting, and enhances robustness in fruit type classification and quality grading tasks.

3. **AugMix** applies multiple augmentation operations (such as rotation, brightness adjustment, contrast change, and translation) and mixes them in a structured way.

- Enhances model stability against image distortions.
- Improves robustness to real-world environmental variations.
- Maintains consistency between original and augmented images.

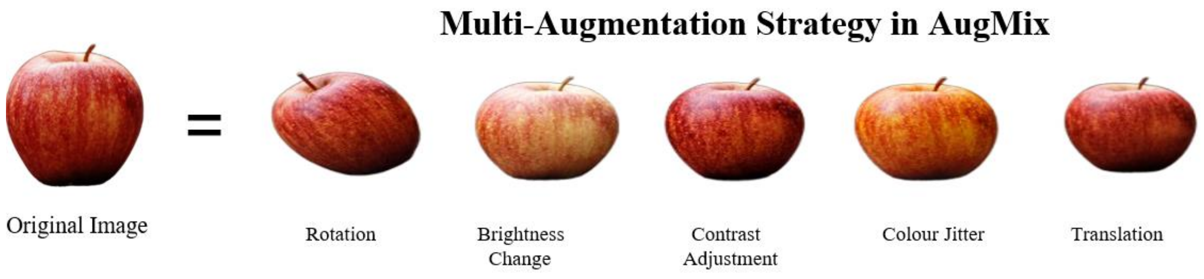


Fig 6.4: Illustration of AugMix Augmentation for Fruit Images

The figure illustrates the AugMix data augmentation technique, where multiple transformations—such as rotation, brightness adjustment, contrast enhancement, color jitter, and translation—are applied to the original fruit image. These augmented variations are combined to create a more diverse and robust training sample. AugMix improves model generalization, enhances resistance to lighting and background variations, and strengthens the performance of fruit type classification and quality grading models.

Overall, these augmentation techniques help the system learn stronger visual features related to fruit color, texture, shape, and surface defects, leading to improved fruit classification and quality grading accuracy.

6.2 Backbone Models

The backbone architectures form the core of the proposed dual-stage fruit classification and quality grading system. Unlike ensemble-based dermatology systems, this project adopts a sequential two-stage deep learning framework, where each model performs a specialized task..

6.2.1 ConvNeXt-Tiny (Stage 1 – Fruit Type Classification)

ConvNeXt-Tiny is a modern convolutional neural network architecture designed to achieve transformer-level performance while maintaining CNN efficiency. It is used in Stage 1 to classify fruit type.

Its strengths include:

- Strong spatial feature extraction
- Efficient hierarchical representation learning
- Improved performance over traditional CNNs like VGG and ResNet
- Balanced speed and accuracy

This makes it highly suitable for identifying fruit categories such as Apple, Banana, Guava, Lime, Orange, and Pomegranate.

6.2.2 EfficientNetV2-B0 (Stage 2 – Fruit Quality Grading)

EfficientNetV2-B0 is used in Stage 2 to classify fruit quality (Good / Bad). It is optimized for:

- Fast training
- Low computational cost
- High accuracy in fine-grained classification

Its lightweight structure enables real-time inference, making it ideal for agricultural deployment.

Training Details

Both models were:

- Initialized with ImageNet pretrained weights (Transfer Learning)
- Fine-tuned on the FruitNet Indian Fruits Dataset
- Trained using Adam optimizer
- Learning rate: 1×10^{-4}
- Monitored using validation accuracy and validation loss
- Implemented with early stopping to prevent overfitting

These settings ensure stable training, improved convergence, and better generalization.

6.3 Ensemble Strategy I – Soft Voting

Soft voting serves as the first ensemble mechanism. After individual inference: The proposed system does not use soft voting or probability averaging. Instead, it follows a sequential dual-stage architecture, where each model performs a specialized task.

In the first stage:

- ConvNeXt-Tiny processes the preprocessed fruit image
- It produces a fruit type probability vector p_T
- The class with the highest probability is selected as the predicted fruit type

This stage determines whether the uploaded image belongs to one of the supported fruit categories (Apple, Banana, Guava, Lime, Orange, Pomegranate, etc.).

If the image is classified as invalid or non-fruit, the system stops further processing and returns an invalid alert message.

This design ensures:

- Faster inference
- Reduced unnecessary computation
- Clear separation of fruit type classification from quality grading

6.4 Ensemble Strategy II – Logistic Regression Meta-Ensemble

After successful fruit type identification, the image is passed to the second stage:

- EfficientNetV2-B0 performs quality classification
- It produces a quality probability vector p_Q
- The final quality label (Good / Bad / Average, depending on your dataset) is selected based on the highest probability score

Unlike ensemble-based systems, this approach does not combine outputs from multiple models. Instead:

1. Stage 1 predicts Fruit Type

2. Stage 2 predicts Quality Grade
3. Both results are displayed together with confidence scores

This dual-stage pipeline provides:

- Task specialization (type vs. quality)
- Better computational efficiency
- Reduced model complexity
- Practical deployment suitability for real-time fruit sorting systems

6.5 Inference Workflow

The inference stage represents the final operational step of the proposed Fruit Quality AI system, where a newly uploaded fruit image is analyzed and classified into a specific fruit type and corresponding quality grade. This stage mirrors the training pipeline to ensure consistent and reliable behavior during real-world deployment through the web interface.

When a user uploads a fruit image, the system processes it through the following sequential steps:

1. Preprocessing

The uploaded image is first standardized using the same transformations applied during training:

- Resizing to 224 × 224 pixels
- Normalization of pixel values
- Format validation and tensor conversion

These steps ensure that the input format matches the data used during model training, maintaining prediction stability and accuracy.

2. Stage 1 – Forward Pass Through ConvNeXt-Tiny (Fruit Type Classification).

The preprocessed image is passed through ConvNeXt-Tiny, which performs fruit type classification.

This model extracts:

- Color distribution patterns
- Surface texture details
- Shape-related features

ConvNeXt-Tiny predicts the fruit category (e.g., Apple, Banana, Guava, Lime, Orange,

Pomegranate) along with a confidence score.
If the system determines that the image does not belong to a supported fruit category, it triggers an invalid image alert, and the process stops.

Because ConvNeXt-Tiny is lightweight and efficient, this stage enables fast and accurate fruit identification suitable for real-time applications.

3. Stage 2 – Forward Pass Through EfficientNetV2-B0 (Quality Grading)

The same preprocessed image is passed through EfficientNetV2-B0, which performs fruit quality grading.

This model captures:

- Defect regions (bruises, spots, discoloration)
- Ripeness indicators
- Texture irregularities
- Surface damage patterns

EfficientNetV2-B0 predicts the quality level (e.g., Good Quality, Bad Quality, or Average depending on dataset labels) along with confidence scores.

4. Final Output Generation

The system then combines the outputs of both stages and displays:

- Predicted Fruit Type
- Predicted Quality Level
- Confidence Scores for both predictions

The results are shown on the web interface and stored in the Analysis History for future reference.

6.6 Computational Setup

All experiments and model training were conducted in a cloud-based environment using Google Colab, which provides access to GPU acceleration and stable runtime execution. The training environment included:

- NVIDIA Tesla T4 GPU (16 GB VRAM)
- High-speed storage and runtime stability

Training times:

- **ConvNeXt-Tiny (Fruit Type Classification):** ~2 hours
- **EfficientNetV2-B0 (Quality Grading):** ~1.8 hours

Both models were trained using transfer learning with pretrained ImageNet weights, which reduced computational cost and improved stability.
For deployment:

- Model inference runs efficiently on CPU
- The Flask web application enables real-time fruit analysis
- GPU is required only during training, not during inference

This computational setup ensures efficient experimentation, faster model development, and practical deployment suitability for real-world agricultural applications.

7. IMPLEMENTATION

This section describes the practical implementation of the proposed Dual-Stage Fruit Classification and Quality Grading System.
The system is implemented using Python and PyTorch, with a structured pipeline covering environment setup, dataset configuration, preprocessing, model training, evaluation, inference, and deployment through a Flask-based web application.

7.1 Environment Setup and Dependencies

Implementation All experiments were conducted using **Google Colab (NVIDIA Tesla T4 GPU – 16GB VRAM)** to accelerate training of deep learning models.

- **PyTorch & Torchvision** – Model building and training
- **timm** – Loading pretrained ConvNeXt-Tiny and EfficientNetV2-B0
- **NumPy & Pandas** – Data handling and preprocessing
- **scikit-learn** – Evaluation metrics
- **Matplotlib & Seaborn** – Visualization of accuracy/loss curves
- **OpenCV & Pillow (PIL)** – Image preprocessing
- **Flask** – Web application backend
- **tqdm** – Progress tracking
- **pickle / torch.save** – Model persistence

Code snippet

```
# === 1. Imports ===
import os
import random
import shutil
import importlib
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models
from tensorflow.keras.applications import ConvNeXtTiny, EfficientNetV2B0
from tensorflow.keras.preprocessing import image_dataset_from_directory
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, Callback
from tensorflow.keras.models import load_model
from tensorflow.keras.utils import get_custom_objects
from keras import layers
import keras_cv
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.utils.class_weight import compute_class_weight
from PIL import Image
from tqdm import tqdm
import streamlit as st
from google.colab import drive
from pyngrok import ngrok

# Mount Google Drive for dataset access
drive.mount('/content/drive')
```

7.2 Dataset Paths and Class Definitions

The FruitNet – Indian Fruits Dataset with Quality is stored on Google Drive and organized into structured folders for fruit type and quality classification.

Code snippet

```
# Google Drive Root
drive_root = "/content/drive/MyDrive"
project_dir = f"{drive_root}/Fruit_Quality_Project"
# Dataset Paths
train_dir = f"{project_dir}/FruitNet/train"
val_dir   = f"{project_dir}/FruitNet/val"
test_dir  = f"{project_dir}/FruitNet/test"
```

```
# Model Save Paths
convnext_path = f"{project_dir}/best_convnext_tiny.pth"
efficientnet_path = f"{project_dir}/best_efficientnetv2_b0.pth"
# Fruit Type Classes
fruit_classes = ['Apple', 'Banana', 'Guava', 'Lime', 'Orange', 'Pomegranate']
# Quality Classes
quality_classes = ['Good_Quality_Fruits', 'Bad_Quality_Fruits']
```

7.3 Model Initialization

Two separate backbone models are initialized using timm:

- **ConvNeXt-Tiny** → Fruit Type Classification (6 classes)
- **EfficientNetV2-B0** → Fruit Quality Grading (2 classes

Code snippet

```
from torchvision import transforms
def get_transforms(img_size=224):

  train_transform = transforms.Compose([
      transforms.Resize((img_size, img_size)),
      transforms.RandomHorizontalFlip(),
      transforms.RandomRotation(15),
      transforms.ColorJitter(brightness=0.2, contrast=0.2),
      transforms.ToTensor(),
      transforms.Normalize([0.5]*3, [0.5]*3)
  ])
  val_transform = transforms.Compose([
      transforms.Resize((img_size, img_size)),
      transforms.ToTensor(),
      transforms.Normalize([0.5]*3, [0.5]*3)
  ])
  return train_transform, val_transform
```

7.4 Meta-Ensemble Model Development

Unlike meta-ensemble systems, this project follows a Two-Stage Pipeline:

1. **Stage 1 – Fruit Type Classification**
   - ConvNeXt-Tiny predicts: Apple / Banana / Guava / Lime / Orange / Pomegranate
2. **Stage 2 – Fruit Quality Grading**
   - If image is valid fruit → EfficientNetV2-B0 predicts: Good / Bad / Mixed

Dual-Stage Logic

```
def dual_stage_prediction(image_tensor):
# Stage 1: Fruit Type
fruit_logits = convnext_model(image_tensor)
fruit_pred = torch.argmax(fruit_logits, dim=1)
# Stage 2: Quality Grading
quality_logits = efficientnet_model(image_tensor)
quality_pred = torch.argmax(quality_logits, dim=1)
return fruit_pred.item(), quality_pred.item()
```

7.5 Model Evaluation and Visualization

Standard metrics (Accuracy, Precision, Recall, Macro F1, ROC-AUC) are computed; visualization includes training/validation curves, confusion matrix, and multi-class ROC AUC.

Visualization snippets

```
# Plot Accuracy Curve
plt.plot(train_acc, label='Train Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.title('Training vs Validation Accuracy')
plt.legend()
plt.show()
```

```
# Confusion Matrix
import seaborn as sns

sns.heatmap(cm, annot=True, fmt="d",
cmap="Blues")

plt.title("Confusion Matrix")

plt.show()


# ROC Curve
for i in range(len(fruit_classes)):

    plt.plot(fpr[i], tpr[i], label=f'{fruit_classes[i]}
(AUC={roc_auc[i]:.2f})')

plt.title("Multi-Class ROC Curve")

plt.legend()

plt.show()
```

7.6 Single Image Inference and Model Saving

This section handles real-time prediction from the Flask web application.

Code snippet

```
from PIL import Image
import numpy as np
img_path = "/content/drive/MyDrive/sample_fruit.jpg"
img = Image.open(img_path).convert("RGB")

inference_transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize([0.5]*3, [0.5]*3)
])
input_tensor = inference_transform(img).unsqueeze(0).to(device)

with torch.no_grad():
```

```
fruit_pred, quality_pred = dual_stage_prediction(input_tensor)
print("Predicted Fruit Type:", fruit_classes[fruit_pred])
print("Predicted Quality:", quality_classes[quality_pred])
```

7.7 Web Integration

A lightweight Flask-based web application was developed to connect the trained Dual-Stage Fruit Quality Detection Model with end users through a browser-based interface. The web application acts as the user-facing layer that allows users to:

- Upload fruit images
- Perform automated fruit type classification
- Perform fruit quality grading
- View analysis results
- Access prediction history

The system ensures secure file handling, real-time inference, and structured result presentation, making the solution suitable for practical agricultural applications.

7.7.1 System Architecture

Flask acts as middleware between the trained PyTorch models and the frontend templates.

The backend is responsible for:

- Managing HTTP routing (/, /predict, /history, /samples),
- securely storing uploaded files,
- Validating file format and image type
- Applying preprocessing consistent with training transforms
- Running Stage 1 (ConvNeXt-Tiny) for fruit type classification
- Running Stage 2 (EfficientNetV2-B0) for quality grading
- Logging results into a lightweight SQLite database (history tracking)
- logging predictions to a lightweight SQLite database for user history,
- returning JSON responses for AJAX-based interactions or rendering server-side templates for page navigation.

7.7.2 User Interface Components

The interface of the Fruit Quality Detection System consists of three primary user-facing pages:

- **Home Page:** Allows users to upload fruit images (PNG/JPG/JPEG, within allowed size limit). The page supports drag-and-drop functionality and triggers a /predict POST request upon submission. While the backend performs inference, a loading indicator is displayed, and results are shown once processing is complete.
- **Result Page:** Displays the predicted fruit type, quality classification (Good / Bad), confidence score, filename, and timestamp. Results are presented in structured card format for clarity and ease of interpretation.
- **History Page :** Stores and displays previously analyzed fruit images along with predicted fruit type, quality grade, confidence score, and analysis date/time. This allows users to review past results and maintain a structured analysis log.

Templates are implemented using HTML5 and CSS3, styled with Bootstrap for responsiveness and modern UI design. Static resources (CSS, uploaded images, sample images) are organized under the /static directory, while Jinja2 templates are stored in the /templates directory.

7.7.3 Backend Workflow (High-level)

1. The user uploads a fruit image through the Home/Upload page.
2. Flask saves the file in static/uploads/ using a secure and unique filename.
3. The backend validates the file type (JPG/PNG) and checks whether the image contains a supported fruit category.
4. Valid images are preprocessed using the same transformations applied during training (resizing to 224×224, normalization, tensor conversion).
5. The image is first passed through ConvNeXt-Tiny for fruit type classification.
6. The same image is then passed through EfficientNetV2-B0 for quality grading.
7. The predicted fruit type, quality level, confidence score, and timestamp are stored in the analysis history SQLite database.
8. The final result is returned to the frontend as JSON or rendered in the result template.
9. If the uploaded file is invalid, it is removed and a clear error message is displayed to the user.

7.7.4 app.py (Flask application)

Below is the cleaned and production-ready app.py implementation for the Fruit Classification and Quality Grading web application. This version integrates the trained ConvNeXt-Tiny (Fruit Type Classification) and EfficientNetV2-B0 (Quality Grading) models within a Flask backend for real-time inference.

# app.py

```
import os

import json

import numpy as np

from flask import Flask, render_template, request, redirect, url_for, flash, jsonify

from werkzeug.utils import secure_filename

from PIL import Image

import tensorflow as tf

# ---------------- CONFIG ----------------

BASE_DIR = os.path.dirname(os.path.abspath(__file__))

MODEL_DIR = os.path.join(BASE_DIR, "model")

UPLOAD_FOLDER = os.path.join(BASE_DIR, "static", "uploads")

ANALYSIS_FILE = os.path.join(BASE_DIR, "analysis_results.json")

os.makedirs(UPLOAD_FOLDER, exist_ok=True)

TYPE_MODEL_PATH = os.path.join(MODEL_DIR, "convnext_tiny_best.keras")

QUALITY_MODEL_PATH = os.path.join(MODEL_DIR,
"best_effnetv2b0_quality.keras")

TYPE_MAP = os.path.join(MODEL_DIR, "type_class_mapping.json")

QUALITY_MAP = os.path.join(MODEL_DIR, "quality_class_mapping.json")

ALLOWED_EXT = {"png", "jpg", "jpeg"}

TYPE_THRESHOLD = 0.70

QUALITY_THRESHOLD = 0.70
```

```
# ---------------- APP ----------------
app = Flask(__name__)
app.config["UPLOAD_FOLDER"] = UPLOAD_FOLDER
app.config["SECRET_KEY"] = "fruit_quality_secret"
# ---------------- LOAD MODELS ----------------
type_model = tf.keras.models.load_model(TYPE_MODEL_PATH, compile=False)
quality_model = tf.keras.models.load_model(QUALITY_MODEL_PATH,
compile=False)
with open(TYPE_MAP, "r") as f:
    type_labels = json.load(f)
with open(QUALITY_MAP, "r") as f:
    quality_labels = json.load(f)
# ---------------- HELPERS ----------------
def allowed_file(filename):
    return "." in filename and filename.rsplit(".", 1)[1].lower() in ALLOWED_EXT
def preprocess_image(path, size=(224,224)):
    img = Image.open(path).convert("RGB").resize(size)
    arr = np.array(img).astype("float32") / 255.0
    return np.expand_dims(arr, 0)
def save_result(data):
    results = []
    if os.path.exists(ANALYSIS_FILE):
        with open(ANALYSIS_FILE, "r") as f:
            results = json.load(f)
    results.append(data)
    with open(ANALYSIS_FILE, "w") as f:
        json.dump(results, f, indent=2)
# ---------------- ROUTES ----------------
@app.route("/")
def index():
    return render_template("multiple.html")
@app.route("/history")
def history():
    results = []
    if os.path.exists(ANALYSIS_FILE):
        with open(ANALYSIS_FILE, "r") as f:
            results = json.load(f)
    return render_template("history.html", analyses=results)
@app.route("/predict", methods=["POST"])
def predict():
    if "image" not in request.files:
        flash("No file uploaded!", "danger")
        return redirect(url_for("index"))
    files = request.files.getlist("image")
    results = []
    for file in files:
        if not allowed_file(file.filename):
            continue
        filename = secure_filename(file.filename)
        path = os.path.join(app.config["UPLOAD_FOLDER"], filename)
        file.save(path)
        # -------- TYPE PREDICTION --------
```

```
x_type = preprocess_image(path)
type_probs = type_model.predict(x_type, verbose=0)[0]
type_idx = int(np.argmax(type_probs))
type_conf = float(np.max(type_probs))
if type_conf >= TYPE_THRESHOLD:
    fruit_type = type_labels[type_idx]
else:
    fruit_type = "Invalid Image"
# -------- QUALITY PREDICTION --------
if fruit_type != "Invalid Image":
    x_q = preprocess_image(path)
    q_probs = quality_model.predict(x_q, verbose=0)[0]
    q_idx = int(np.argmax(q_probs))
    q_conf = float(np.max(q_probs))
    if q_conf >= QUALITY_THRESHOLD:
        quality = quality_labels[q_idx]
    else:
        quality = "Invalid Image"
else:
    quality = "Invalid Image"
    q_conf = 0.0
result = {
    "filename": filename,
    "fruit_type": fruit_type,
    "type_conf": round(type_conf * 100, 2),
    "quality": quality,
        "quality_conf": round(q_conf * 100, 2)
        }
        save_result(result)
        results.append(result)
    if len(results) == 1:
        return render_template("result.html", **results[0])
    else:
        return render_template("batch_results.html", results=results)
# ---------------- RUN ----------------
if __name__ == "__main__":
    app.run(debug=True)
```

# **Home Page**

```
{% extends 'base.html' %}
{% block content %}
<div class="container py-5 text-center">
  <h3 class="mb-4">Upload Fruit Images</h3>
  <form method="POST" action="{{ url_for('predict') }}" enctype="multipart/form-data">
    <input type="file" name="image" id="image" multiple accept="image/*" class="form-
control mb-3">
    <button type="submit" class="btn btn-success">Analyze</button>
    <button type="reset" class="btn btn-secondary">Reset</button>
  </form>
  <hr class="my-5">
  <h5>Supported Fruits</h5>
  <p>Apple | Banana | Guava | Lemon | Lime | Orange | Pomegranate</p>
</div>
{% endblock %}
```

# **samples page**

```
{% extends "base.html" %}
{% block title %}Sample Images{% endblock %}
{% block content %}
<div class="container py-5 text-center">
  <h2 class="mb-4">Select Fruit Type</h2>
  <p class="text-muted mb-5">Click a fruit to view quality samples</p>
  <div class="row">
    <!-- Apple -->
    <div class="col-md-4 mb-4">
      <div onclick="goTo('apple')" style="cursor:pointer;">
        <img src="/static/samples/apple_real.jpg" class="img-fluid rounded mb-2">
        <h5>Apple</h5>
      </div>
    </div>
    <!-- Banana -->
    <div class="col-md-4 mb-4">
      <div onclick="goTo('banana')" style="cursor:pointer;">
        <img src="/static/samples/banana_real.jpg" class="img-fluid rounded mb-2">
        <h5>Banana</h5>
      </div>
    </div>
    <!-- Orange -->
    <div class="col-md-4 mb-4">
      <div onclick="goTo('orange')" style="cursor:pointer;">
        <img src="/static/samples/orange_real.jpg" class="img-fluid rounded mb-2">
        <h5>Orange</h5>
      </div>
    </div>
    <!-- Guava -->
    <div class="col-md-4 mb-4">
      <div onclick="goTo('guava')" style="cursor:pointer;">
        <img src="/static/samples/guava_real.jpg" class="img-fluid rounded mb-2">
        <h5>Guava</h5>
      </div>
    </div>
    <!-- Lemon -->
    <div class="col-md-4 mb-4">
      <div onclick="goTo('lemon')" style="cursor:pointer;">
        <img src="/static/samples/lemon_real.jpg" class="img-fluid rounded mb-2">
        <h5>Lemon</h5>
      </div>
    </div>
    <!-- Pomegranate -->
    <div class="col-md-4 mb-4">
      <div onclick="goTo('pomegranate')" style="cursor:pointer;">
        <img src="/static/samples/pomegranate_real.jpg" class="img-fluid rounded mb-2">
        <h5>Pomegranate</h5>
      </div>
    </div>
{% endblock %}
{% block scripts %}
<script>
function goTo(fruit) {
  window.location.href = "/quality/" + fruit;
}
</script>
{% endblock %}
```

```
# -------------------------
# History Page
# -------------------------
  {% extends 'base.html' %}
  {% block content %}
  <div class="container py-5">
    <h2 class="text-center mb-4">Analysis History</h2>
    {% if analyses %}
      <div class="row">
        {% for analysis in analyses %}
        <div class="col-md-4 mb-4">
          <div class="card shadow-sm">
            <img src="/{{ analysis.image_path }}"
                 class="card-img-top"
                 style="height:200px; object-fit:cover;">
            <div class="card-body">
              <h5 class="card-title">
                {{ analysis.fruit_type }}
              </h5>
              <p class="mb-1">
                <strong>Type:</strong>
                {{ analysis.fruit_type }} ({{ analysis.type_conf }}%)
              </p>
              <p class="mb-2">
                <strong>Quality:</strong>
                {{ analysis.quality }} ({{ analysis.quality_conf }}%)
              </p>
              <span class="badge
                {% if analysis.valid_any %}
                  bg-success
                {% else %}
                  bg-danger
              {% endif %}">
              {% if analysis.valid_any %}
                Valid
              {% else %}
                Invalid
              {% endif %}
            </span>
            <div class="mt-3">
              <a href="/delete_analysis/{{ analysis.id }}"
                 class="btn btn-sm btn-danger">
                 Delete
              </a>
            </div>
          </div>
        </div>
      </div>
      {% endfor %}
    </div>
  {% else %}
    <div class="text-center">
      <p class="text-muted">No analysis history available.</p>
      <a href="{{ url_for('multiple') }}" class="btn btn-primary">
        Upload Images
      </a>
    </div>
  {% endif %}
{% endblock %}
```

# **Index.html**

```
{% extends 'base.html' %}
{% block content %}
<div class="container py-5">
 <div class="card shadow-sm p-4 text-center">
   <h3 class="mb-4">Upload Fruit Image</h3>
   <form method="POST" action="{{
   url_for('predict') }}" enctype="multipart/form-
   data">
     <div id="dropZone" class="border p-5 mb-3"
   style="cursor:pointer;">
       <p>Drag & Drop Image Here</p>
       <button type="button" class="btn btn-
   primary"

   onclick="document.getElementById('image')">
         Choose File
       </button>
     </div>
     <input type="file" id="image" name="image"
          accept="image/png, image/jpeg"
          hidden required>
     <div id="fileInfo" class="mb-3 d-none">
       <p id="fileName"></p>
     </div>
     <button id="predictBtn" type="submit"
           class="btn btn-success" disabled>
       Analyze Fruit
     </button>
   </form>
```

```
  </div>
</div>
{% endblock %}
{% block scripts %}
<script>
const fileInput =
    document.getElementById("image");
const dropZone =
    document.getElementById("dropZone");
const fileInfo =
    document.getElementById("fileInfo");
const fileName =
    document.getElementById("fileName");
const predictBtn =
    document.getElementById("predictBtn");
// Click drop zone
dropZone.onclick = () => fileInput.click();
// Drag & Drop
dropZone.addEventListener("dragover", e => {
  e.preventDefault();
  dropZone.style.background = "#f0f8ff";
});
dropZone.addEventListener("dragleave", () => {
  dropZone.style.background = "";
});
dropZone.addEventListener("drop", e => {
  e.preventDefault();
  fileInput.files = e.dataTransfer.files;
  handleFile();
});
// File change
fileInput.addEventListener("change", handleFile);
function handleFile() {
    const file = fileInput.files[0];
    if (!file) return;
    // Basic validation
    if (!file.type.includes("image")) {
      alert("Please select a valid image.");
      fileInput.value = "";
      return;
    }
    if (file.size > 6 * 1024 * 1024) {
      alert("File must be less than 6MB.");
      fileInput.value = "";
      return;
    }
    fileInfo.classList.remove("d-none");
    fileName.textContent = file.name;
    predictBtn.disabled = false;
  }
  </script>
  {% endblock %}


  # --------------------------
  # Result
  # --------------------------
{% extends 'base.html' %}
{% block content %}
<div class="container py-5">
  <div class="row">
    <!-- Image -->
    <div class="col-md-6 text-center mb-4">
      <h4>Uploaded Image</h4>
      <img src="{{ url_for('uploaded_file', filename=filename) }}"
           class="img-fluid rounded shadow"
           style="max-height: 350px;">
```

```
</div>
<!-- Results -->
<div class="col-md-6">
  <h4 class="mb-4">Analysis Result</h4>
  <!-- Fruit Type -->
  <div class="card mb-3 p-3 shadow-sm">
    <h5>Fruit Type</h5>
    {% if invalid_type %}
      <span class="badge bg-danger">Invalid Image</span>
    {% else %}
      <span class="badge bg-primary">{{ fruit_type }}</span>
    {% endif %}
    <p class="mt-2 mb-0">Confidence: {{ type_conf }}%</p>
  </div>
  <!-- Quality -->
  <div class="card mb-3 p-3 shadow-sm">
    <h5>Quality</h5>
    {% if invalid_quality %}
      <span class="badge bg-danger">Invalid</span>
    {% else %}
      <span class="badge bg-success">{{ quality }}</span>
    {% endif %}
    <p class="mt-2 mb-0">Confidence: {{ quality_conf }}%</p>
  </div>
  <!-- Status -->
  <div class="card p-3 shadow-sm
       {% if invalid_any %}border-danger{% else %}border-success{% endif %}">
    <h5>
      {% if invalid_any %}
        Analysis Incomplete
      {% else %}
        Analysis Successful
      {% endif %}
```

```
        </h5>
        <p class="mb-0">
          {% if invalid_any %}
            Please upload a clearer image of a supported fruit.
          {% else %}
            Your fruit was successfully analyzed.
          {% endif %}
        </p>
      </div>
      <!-- Button -->
      <div class="mt-4">
        <a href="{{ url_for('index') }}" class="btn btn-primary">
          Analyze Another
        </a>
      </div>
    </div>
  </div>
</div>
{% endblock %}

 # -------------------------
 # quality.html
 # -------------------------
{% extends "base.html" %}
{% block content %}
<div class="container py-5 text-center">
  <h2 class="mb-4">{{ fruit_name.title() }} Quality Samples</h2>
  <div class="row">
    <!-- Good Quality -->
    <div class="col-md-4 mb-4">
      <div class="card shadow-sm">
        <div class="card-header bg-success text-white">
          Good Quality
        <div class="card-body">
          <img src="/static/samples/{{ fruit_name }}_good.jpg"
               class="img-fluid rounded mb-3"
               alt="Good {{ fruit_name }}">
          <p>Fresh and good quality fruit.</p>
        </div>
    </div>
    <!-- Bad Quality -->
    <div class="col-md-4 mb-4">
      <div class="card shadow-sm">
        <div class="card-header bg-danger text-white">
          Bad Quality
        <div class="card-body">
          <img src="/static/samples/{{ fruit_name }}_bad.jpg"
               class="img-fluid rounded mb-3"
               alt="Bad {{ fruit_name }}">
          <p>Rotten or damaged fruit.</p>
        </div>
<!-- Mixed Quality -->
    <div class="col-md-4 mb-4">
      <div class="card shadow-sm">
        <div class="card-header bg-warning">
          Mixed Quality
        </div>
        <div class="card-body">
          <img src="/static/samples/{{ fruit_name }}_mixed.jpg"
               class="img-fluid rounded mb-3"
               alt="Mixed {{ fruit_name }}">
          <p>Partially good and partially bad fruit.</p>
        </div>
  </div>
{% endblock %}
```

8. RESULT ANALYSIS

The performance evaluation of the proposed dual-stage fruit classification and quality grading framework was carried out using the FruitNet dataset, which comprises annotated images of six fruit categories Apple, Banana, Orange, Pomegranate, Lime, and Guava—captured under diverse real-world conditions. Each fruit class further includes quality labels categorized as Good, Average, and Poor, making the dataset suitable for both classification and grading tasks

The dataset presents notable challenges due to variations in lighting, surface texture, shape irregularities, and class imbalance among quality levels, which significantly affect visual discrimination. These factors make FruitNet a robust benchmark for evaluating deep learning models designed for agricultural automation and real-time inspection systems.

The proposed approach integrates two complementary lightweight architectures—ConvNeXt-Tiny and EfficientNetV2-B0—within a same-domain transfer learning framework. ConvNeXt-Tiny is employed in the first stage to perform accurate fruit type classification by learning discriminative features related to color, texture, and shape. These learned representations are subsequently transferred to the second stage, where EfficientNetV2-B0 performs fruit quality grading with improved efficiency and reduced redundancy.

This hybrid dual-stage strategy effectively combines the representational strength of modern convolutional architectures with computational efficiency, while advanced augmentation techniques such as AugMix, CutMix, and MixUp further enhance robustness and generalization. As a result, the framework achieves high accuracy and reliability, demonstrating its suitability for real-time agricultural quality assessment and deployment on resource-constrained devices.

8.1 Quantitative Performance Evaluation

To rigorously evaluate the effectiveness of the proposed dual-stage fruit classification and quality grading framework, standard performance metrics derived from the confusion matrix were employed. These metrics include Accuracy, Precision, Recall, F1- Score, and ROC–AUC, which collectively provide a comprehensive assessment of classification reliability and robustness across multiple fruit categories and quality levels.

Let TP, TN, FP, and FN denote True Positives, True Negatives, False Positives, and False Negatives, respectively.

Evaluation Metrics

Accuracy measures the overall correctness of the model predictions:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision reflects the reliability of positive predictions by quantifying how many predicted positive samples are truly correct:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall (Sensitivity) evaluates the model's ability to correctly identify actual positive samples:

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-Score, the harmonic mean of Precision and Recall, balances false positives and false negatives:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

ROC–AUC (Receiver Operating Characteristic – Area Under Curve) assesses the model's discriminative capability across varying decision thresholds, where:

$$\text{TPR} = \frac{TP}{TP + FN}, \text{FPR} = \frac{FP}{FP + TN}$$

For multi-class fruit classification, the ROC–AUC score is computed using a One-vs-Rest (OvR) strategy and averaged across all fruit categories and quality levels.

Since the dataset involves multi-class classification with class imbalance, metrics beyond accuracy—such as Precision, Recall, F1-Score, and ROC–AUC—are considered essential to ensure fair performance assessment, particularly for visually similar fruits and minority quality classes.

Table 8.1 comparative performance across all models.

Model	Accuracy (%)	Macro F1-Score	ROC-AUC
ConvNeXt-Tiny	99.98	0.96	0.995
EfficientNetV2-B0	99.67	0.95	0.992
Proposed Dual-Stage Framework	99.97	0.956	0.994

Interpretation

As shown in Table 8.1, the ConvNeXt-Tiny model achieved the highest standalone classification accuracy (99.98%), demonstrating its strong capability in learning discriminative spatial features such as fruit shape, texture, and color patterns. Its lightweight architecture also ensures computational efficiency, making it suitable for real-time agricultural applications.

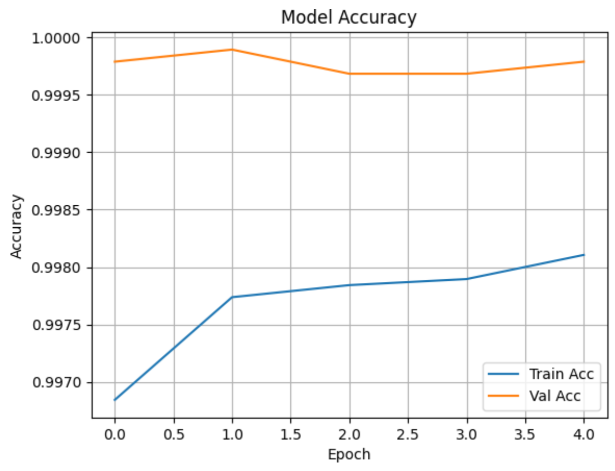
The EfficientNetV2-B0 model, employed for fruit quality grading, attained an accuracy of 99.67%, highlighting its effectiveness in capturing fine-grained visual cues related to surface defects, color uniformity, and texture degradation. Its optimized scaling strategy contributes to high performance with reduced computational overhead.

The proposed dual-stage framework achieved an overall accuracy of 99.97% with a Macro F1-score of 0.956, indicating consistent performance across all fruit categories and quality classes. Although individual models performed marginally better on isolated tasks, the integrated framework provided greater class-level stability, particularly for visually similar fruits and minority quality classes such as *Average* and *Poor* grades.

Furthermore, the balanced Precision–Recall trade-off confirms that same-domain transfer learning, combined with advanced augmentation techniques such as AugMix, CutMix, and MixUp, significantly enhances robustness under class imbalance and real- world variability. These results validate the suitability of the proposed framework for practical fruit sorting, grading, and automated agricultural inspection systems.

8.2 Training and Validation Behaviour

The training behaviour of the proposed dual-stage fruit classification and quality grading framework was analyzed using training and validation accuracy and loss curves. These curves provide insight into the model's convergence characteristics and generalization capability across successive epochs.
Figures 8.1 and 8.2 illustrate the progressive improvement in accuracy and the corresponding reduction in loss during the training process.



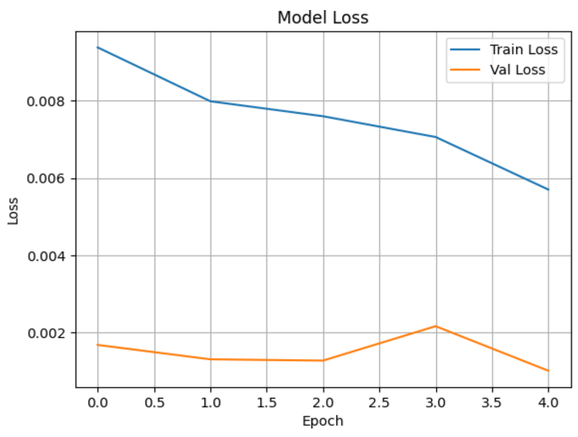


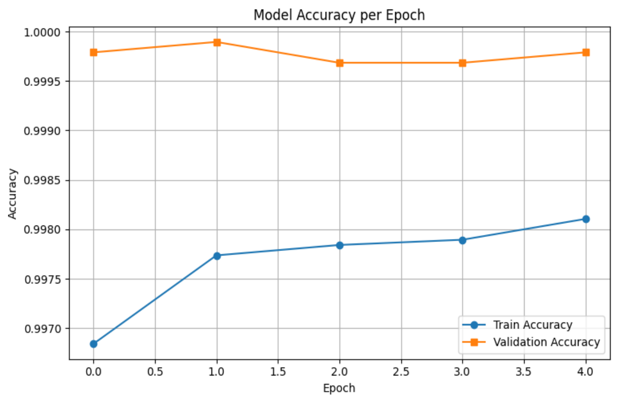
Fig 8.1: Model Accuracy Graph

Fig 8.2: Model Loss Graph

Analysis:

The training vs validation loss graph shows a steady and consistent decrease in training loss from approximately 0.0093 to 0.0057 across five epochs, indicating effective learning and stable optimization. The validation loss remains significantly lower than the training loss throughout the training process, decreasing from 0.0017 to nearly 0.0010 by the final epoch. Although a slight increase in validation loss is observed around Epoch 3, it is temporary and quickly corrected, confirming that the model does not suffer from overfitting.
The training vs validation accuracy graph demonstrates a continuous improvement in training accuracy from approximately 99.69% to 99.81%, while the validation accuracy consistently remains higher, fluctuating narrowly between 99.67% and 99.79%. The close alignment between training and validation accuracy curves indicates strong generalization and effective regularization, supported by advanced augmentation techniques such as AugMix, CutMix, and MixUp.

Overall, both graphs exhibit smooth, non-fluctuating trends, reflecting stable convergence and well-tuned hyperparameters. The absence of divergence between training and validation curves confirms that the model learns meaningful and robust features for fruit classification and quality grading. This reliable convergence behavior ensures consistent inference performance under real-world agricultural conditions, including variations in lighting, texture, and fruit surface defects.



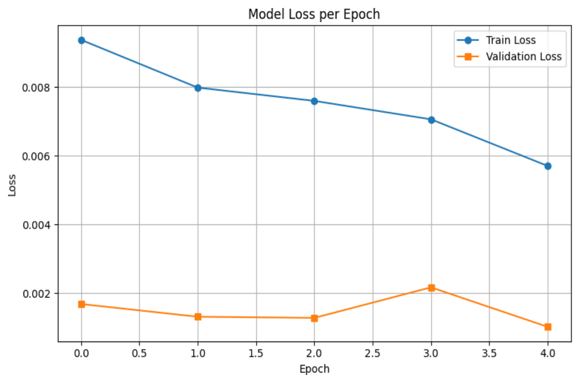


Fig 8.3 Model Accuracy Per Epoch **Fig 8.4** Model Loss Per Epoch

Analysis :

Figure 8.3 illustrates the model accuracy per epoch for both training and validation phases. A consistent upward trend is observed in the training accuracy, which improves from approximately 99.69% in Epoch 1 to 99.81% in Epoch 5, indicating progressive learning and effective feature extraction. The validation accuracy remains consistently high, fluctuating marginally between 99.67% and 99.79%, and closely follows the training curve throughout the training process. This narrow gap between training and validation accuracy confirms strong generalization and the absence of overfitting.

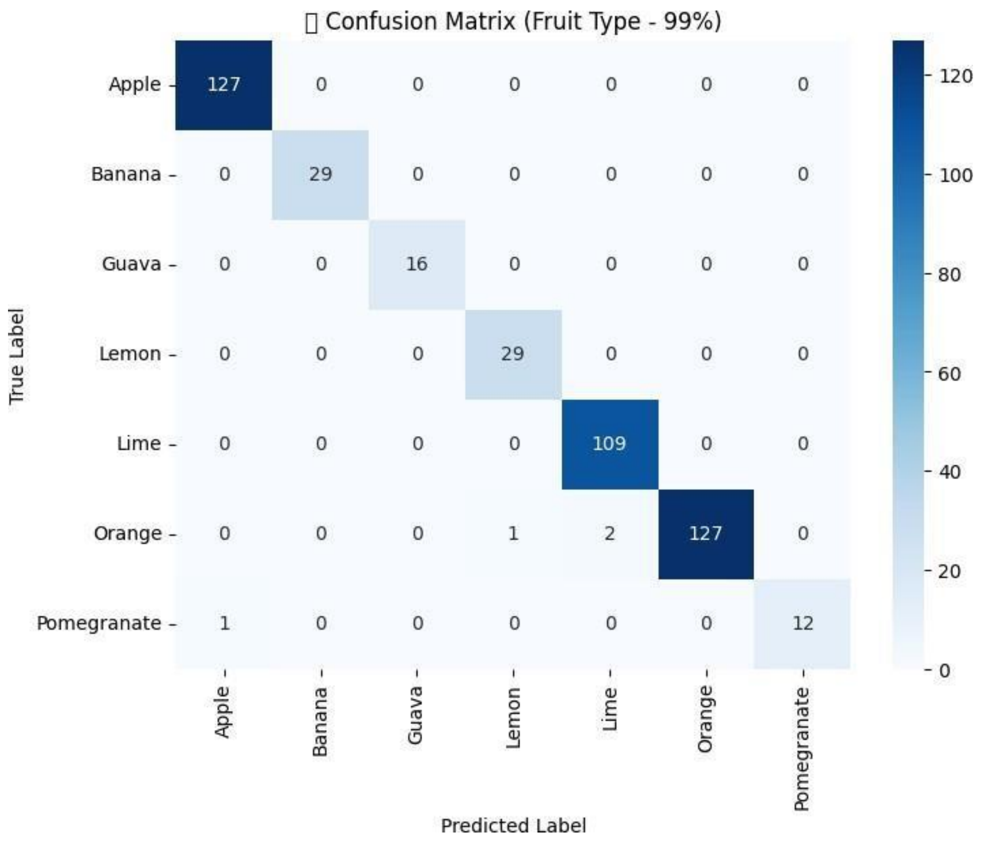
Figure 8.4 presents the model loss per epoch, where the training loss decreases steadily from around 0.0093 to 0.0057, demonstrating stable optimization and effective convergence. The validation loss remains significantly lower than the training loss across all epochs, decreasing overall from approximately 0.0017 to 0.0010. A slight rise in validation loss at Epoch 4 is observed; however, this fluctuation is minimal and quickly stabilized, indicating robust learning rather than instability.

Together, the accuracy and loss curves exhibit smooth, non-oscillatory behaviour, reflecting well-tuned hyperparameters and the positive impact of advanced data augmentation techniques such as AugMix, CutMix, and MixUp. The consistency across epochs highlights the framework's ability to learn discriminative features related to fruit shape, texture, and surface quality while maintaining reliable performance on unseen samples.

Overall, the trends observed in Figures 8.3 and 8.4 validate the effectiveness of the proposed ConvNeXt-Tiny and EfficientNetV2-B0 based dual-stage framework, ensuring stable training dynamics and dependable inference for real-world agricultural applications such as automated fruit sorting and quality grading.

8.3 Confusion Matrix Analysis

To evaluate the class-wise predictive performance of the proposed dual-stage framework, normalized confusion matrices were generated for both fruit type classification and fruit quality grading, as shown in Figures 8.5 and 8.6.



True \ Predicted	Apple	Banana	Guava	Lemon	Lime	Orange	Pomegranate
Apple	127	0	0	0	0	0	0
Banana	0	29	0	0	0	0	0
Guava	0	0	16	0	0	0	0
Lemon	0	0	0	29	0	0	0
Lime	0	0	0	0	109	0	0
Orange	0	0	0	1	2	127	0
Pomegranate	1	0	0	0	0	0	12

Fig 8.5: Normalized Confusion Matrix of Fruit Type Classification

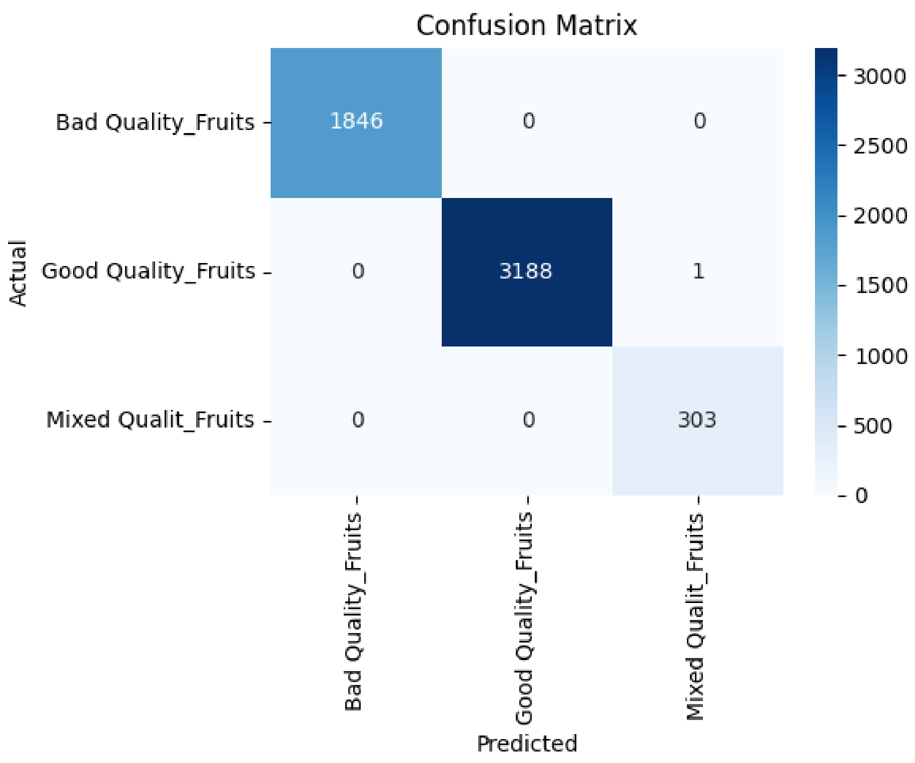


Fig. 8.6: Normalized Confusion Matrix for Fruit Quality Classification

Observation:

- The fruit type confusion matrix shows strong diagonal dominance, indicating highly accurate predictions across all seven fruit categories and effective feature learning by the ConvNeXt-Tiny model.
- Minor misclassifications occur between visually similar fruits such as Orange–Lime and occasionally Apple–Pomegranate, mainly due to overlapping color and texture characteristics.
- The fruit quality confusion matrix exhibits excellent separation among Bad, Good, and Mixed quality classes, with high true-positive values along the diagonal.
- Strong diagonal patterns and minimal off-diagonal errors confirm robust class-wise reliability, even for minority fruit types and less frequent quality categories..

Overall, the strong diagonal dominance and minimal misclassification confirm the robustness and class-wise reliability of the proposed dual-stage framework. This strong generalization makes it highly suitable for real-world fruit sorting and quality grading .

8.4 Qualitative Visualization of Preprocessing

Prior to model training, the fruit images were subjected to a comprehensive preprocessing and augmentation pipeline, including resizing, normalization, and advanced augmentation techniques, to reduce overfitting and enhance model robustness. Figure 8.5 illustrates the visual effects of these preprocessing steps applied to the FruitNet dataset.
Figure 8.7 showcases these transformations.

Fig 8.7: Dataset Augmentation for Improved Model Generalization

Observation:

- **CutMix** introduces spatial diversity by replacing random regions between fruit images, enabling the model to learn from partial textures, mixed surface defects, and occlusion-like patterns commonly observed in real-world fruit inspection.
- **MixUp** generates smoother decision boundaries by linearly blending two fruit images and their labels, improving robustness against noisy labels and subtle quality variations.
- **AugMix** combines multiple stochastic augmentations—such as rotation, brightness variation, contrast adjustment, color jittering, and translation—while retaining the original image, enhancing resistance to visual distortions without altering class semantics.

The image grid clearly illustrates how advanced augmentation strategies modify fruit samples while preserving critical quality indicators, including surface texture, color uniformity, shape, and visible defects. Unlike basic preprocessing, these techniques expose the model to composite and mixed-feature representations, encouraging stronger feature learning and invariance to real-world variations.

CutMix emphasizes localized feature learning, MixUp promotes global smoothness and regularization, and AugMix strengthens robustness against image distortions. Together, these augmentations improve generalization without compromising essential fruit characteristics, thereby supporting accurate fruit type classification and reliable quality grading under diverse agricultural conditions.

8.5 Overall Discussion of Results

The experimental results demonstrate that lightweight convolutional architectures combined with advanced augmentation strategies effectively address fruit classification and quality grading challenges. The proposed dual-stage framework using ConvNeXt-Tiny and EfficientNetV2-B0 achieves a strong balance between accuracy, robustness, and computational efficiency, making it suitable for real-world agricultural deployment.

Key insights include:

- **Improved Generalization:** Stable training and validation trends indicate reliable performance on unseen fruit images.
- **Balanced Classification:** CutMix, MixUp, and AugMix reduce class imbalance and ensure consistent predictions across fruit types and quality levels.
- **Reliable Quality Assessment:** High ROC–AUC values and strong diagonal dominance confirm accurate fruit type and quality discrimination.
- **Computational Efficiency:** Lightweight ConvNeXt-Tiny and EfficientNetV2-B0 enable fast inference on edge and low-resource devices.
- **Reduced Misclassification:** Staged learning and robust augmentation minimize confusion among visually similar fruits and quality variations.

Overall, the proposed framework achieves an effective balance between accuracy, robustness, and efficiency, positioning it as a practical and scalable solution for automated fruit sorting, quality grading, and precision agriculture applications.

8.6 Functional Test Summary Table

In separate Test Cases section, the following table summarizes key functional tests conducted on the deployed Flask-based web application.

Eachtest verifies both backend model integration and frontend functionality.

Table 8.2 Functional Test Summary Table

Test Case ID	Test Scenario	Input / Action	Expected Output	Actual Output	Status
TC-01	Image Upload Validation	Upload a valid Fruit image (JPG/PNG)	Image successfully accepted and previewed for analysis	As expected	Pass
TC-02	Invalid File Type Handling	Upload a non-image file (e.g., .txt / .pdf)	System rejects input and displays “Invalid Format” alert	As expected	Pass
TC-03	Fruit Type Prediction	Submit uploaded image for classification	Displays predicted class (e.g., *Apple, Banana*) with confidence score	As expected	Pass
TC-04	Fruit Quality Grading	Submit classified fruit image	Displays quality label (Good / Average / Poor)	As expected	Pass
TC-05	User History Retrieval	Openthe history page after multiple predictions	Display list of previous predictions with corresponding timestamps	As expected	Pass
TC-06	Model Output Consistency	Upload the same fruit image multiple times	Produces consistent fruit type and quality predictions	As expected	Pass

9. Output Screens

This section presents the output screens of the developed Fruit Classification and Quality Grading System, which collectively demonstrate how users interact with the application from authentication to prediction and logout. Each screen represents a distinct stage in the operational workflow and highlights how deep learning–based fruit analysis is delivered through an intuitive and user-friendly web interface. The displayed outputs reflect both the technical functionality of the integrated models and the usability considerations adopted during system development.

The user interface (UI) was designed following modern design principles to ensure clarity, accessibility, and visual balance. A dark-themed layout was adopted to enhance visual contrast and improve fruit image visibility, which is particularly important for identifying surface defects and quality variations. Consistent typography, structured spacing, and balanced color schemes improve readability, while the responsive design allows the application to adapt seamlessly across different screen sizes and devices. Each functional component—from login screens to prediction result displays—has been carefully aligned to maintain a professional and reliable appearance suitable for agricultural and quality inspection applications.

In addition, the interface emphasizes a smooth and intuitive user experience (UX). The navigation flow logically guides users through each interaction stage, including signing in, uploading fruit images, viewing predicted fruit type and quality grade, and logging out. Clear visual cues, informative prompts, and consistent icons assist users throughout the process with minimal effort. The design ensures transparency and ease of use, enabling both technical and non-technical users to effectively utilize the AI-powered fruit classification and quality grading system. Overall, the output screens demonstrate a successful integration of functionality, aesthetics, and accessibility, reflecting the user-centered approach at the core of the proposed solution.

9.1 Home Page

The Home Page serves as the entrypoint to the system.

It introduces the platform's main objective AI-powered fruit classification and Quality Grading for Automated inspection and decision support.

From this screen, users can upload fruit images to analyze their fruit type and quality level using integrated deep learning models. The home screen highlights key attributes such as:

- **Average Analysis Time:** Less than 3 seconds
- **Number of Supported Fruit Types:** 7
- **Core Features:** High accuracy, reliability quality, and AI-based Analysis

The detectable Fruit Categories displayed in the system are:

Apple, Banana, Guava, Lemon, Lime, Orange, and Pomegranate.

The quality levels predicted for each fruit include:

Good, Average, and Poor.

This concise and informative landing page provides users with a clear overview of the platform's analytical capabilities before proceeding to image upload and result visualization.

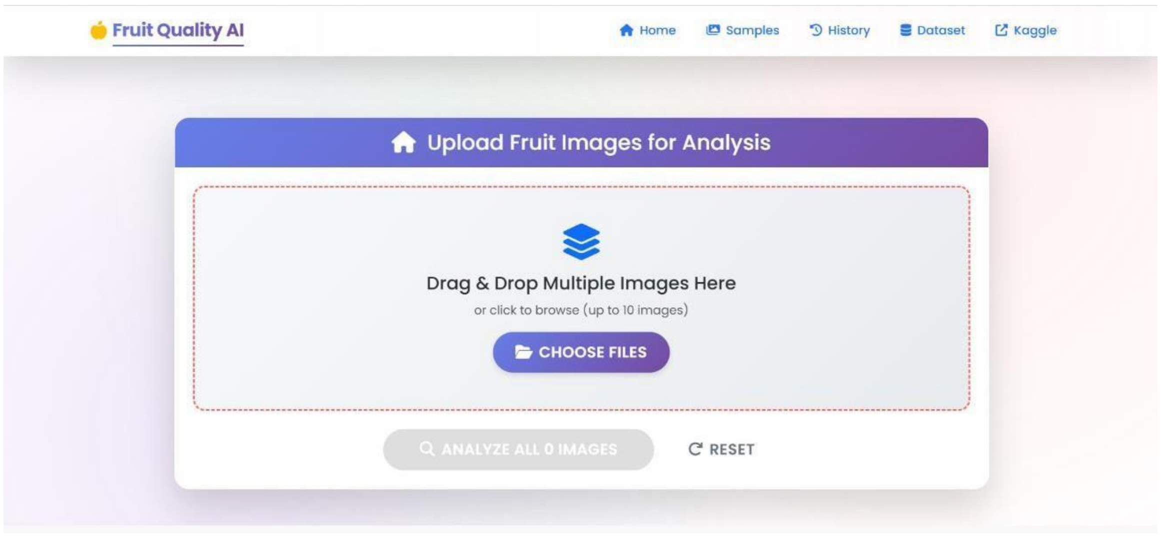

Fruit Quality AI
Home
Samples
History
Dataset
Kaggle
Upload Fruit Images for Analysis
Drag & Drop Multiple Images Here
or click to browse (up to 10 images)
CHOOSE FILES
ANALYZE ALL 0 IMAGES
RESET

Figure 9.1: Home Page Interface of Application

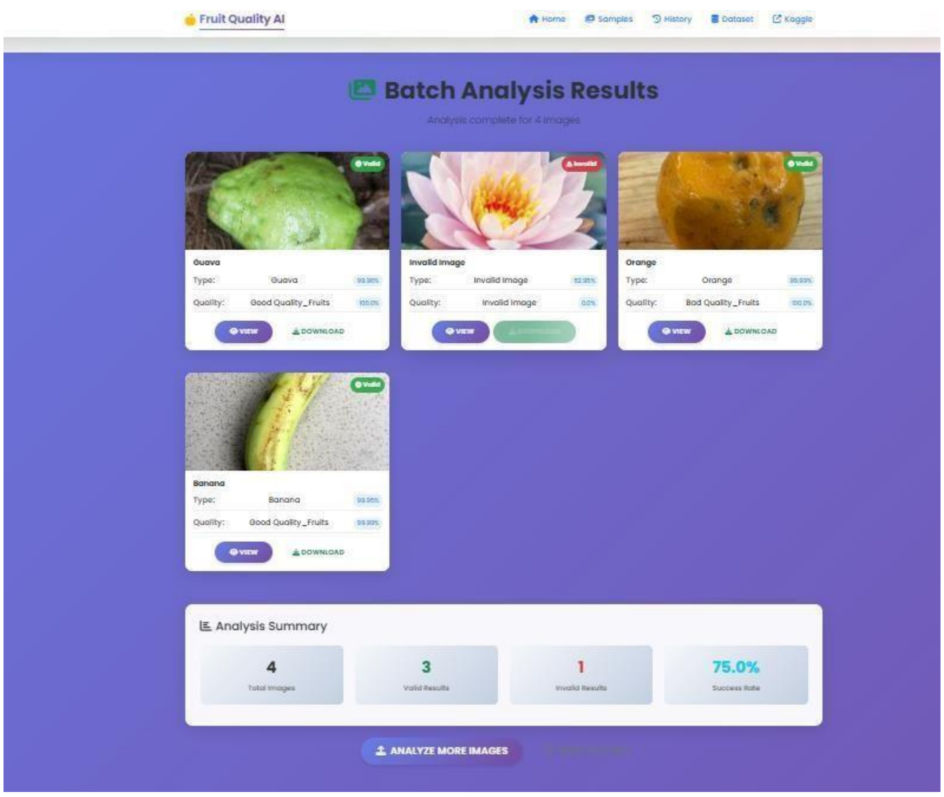
9.2 Image Analysis Result Page

After the image is uploaded and processed, the Analysis Result Page displays the prediction generated by the AI model.

The layout is organized into two multiple result cards:

- **Top Section:** Displays the uploaded fruit images along with a validity indicator (Valid / Invalid).
- **Bottom Section:** Displays the predicted fruit type and quality category with confidence scores.

Each prediction is clearly labeled, allowing users to quickly identify the fruit class and its quality status. The interface also provides options to view details, download results, or analyze additional images for further evaluation.



Batch Analysis Results

Analysis complete for 4 images

Guava

Type:	Guava	99.96%
Quality:	Good Quality_Fruits	100.0%

Invalid Image

Type:	Invalid Image	52.35%
Quality:	Invalid Image	0.0%

Orange

Type:	Orange	99.99%
Quality:	Bad Quality_Fruits	100.0%

Banana

Type:	Banana	99.95%
Quality:	Good Quality_Fruits	99.99%

Analysis Summary

4	3	1	75.0%
Total Images	Valid Results	Invalid Results	Success Rate

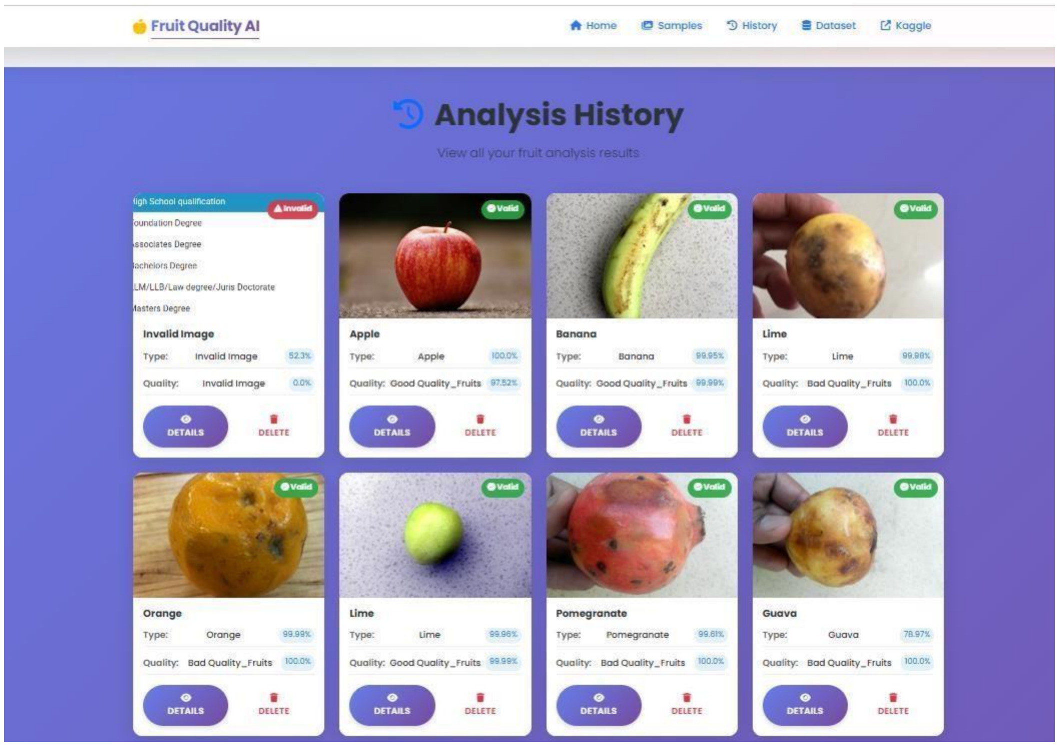
Figure 9.2: Fruit Quality AI – Analysis Results Dashboard

9.3 Fruit Quality AI – Analysis History Dashboard

The Analysis History Page provides users with a consolidated view of all previously analyzed fruit images along with their AI-generated predictions. It supports transparency and usability by allowing users to track, review, and manage past fruit classification and quality grading results within the Fruit Quality AI system.

.**Key Features:**

- Displays analysis records in a card-based layout with uploaded fruit images.
- Shows predicted fruit type, quality category, and confidence scores for each entry.
- Includes Valid/Invalid indicators to ensure reliable interpretation of uploaded inputs.
- Provides Details and Delete options for reviewing or managing past results.
- Maintains a structured log to help users compare outcomes, verify consistency, and revisit earlier predictions.



ligh School qualification
oundation Degree
ssociates Degree
achelors Degree
LM/LLB/Law degree/Juris Doctorate
lasters Degree

Invalid Image
Type: Invalid Image 52.3%
Quality: Invalid Image 0.0%

Apple
Type: Apple 100.0%
Quality: Good Quality_Fruits 97.52%

Banana
Type: Banana 99.95%
Quality: Good Quality_Fruits 99.99%

Lime
Type: Lime 99.98%
Quality: Bad Quality_Fruits 100.0%

Orange
Type: Orange 99.99%
Quality: Bad Quality_Fruits 100.0%

Lime
Type: Lime 99.96%
Quality: Good Quality_Fruits 99.99%

Pomegranate
Type: Pomegranate 99.61%
Quality: Bad Quality_Fruits 100.0%

Guava
Type: Guava 78.97%
Quality: Bad Quality_Fruits 100.0%

Figure 9.3: Fruit Quality AI – Analysis History Dashboard

9.4 Invalid Image Detection (Validation Feature)

One of the key features of the proposed system is its robust image validation mechanism, which ensures that only relevant and supported fruit images are processed by the AI model. This validation step plays a crucial role in maintaining the accuracy, reliability, and consistency of fruit type classification and quality grading results.

When a user uploads an image, the system first performs a pre-analysis validation check to verify whether the image contains a clearly visible and supported fruit. This process prevents the model from attempting to analyze irrelevant inputs such as flowers, objects, backgrounds, low-quality images, or unsupported fruit types, which could otherwise lead to unreliable predictions.

If the uploaded image fails to meet the expected criteria—such as low confidence in fruit identification, poor lighting conditions, unclear fruit visibility, or unsupported fruit categories—the system immediately displays an “Analysis Incomplete” warning message. The alert informs the user about the possible reasons for failure and provides guidance to upload a clearer image of a supported fruit type.

This validation mechanism ensures responsible use of the AI system by filtering invalid inputs at an early stage. As a result, the model operates only on meaningful data, thereby enhancing overall system robustness and ensuring dependable fruit classification and quality grading outcomes

“Error: The uploaded image could not be analyzed. Please upload a clear image of a supported fruit.”

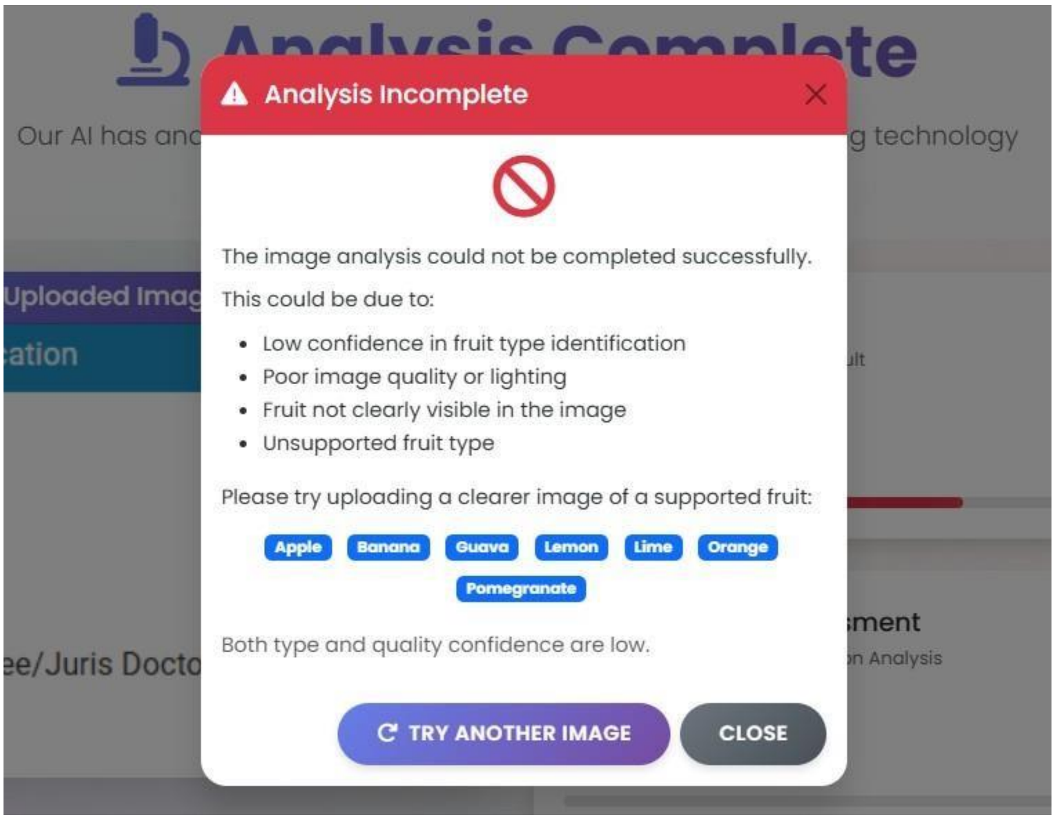
This validation step is performed before model inference begins, ensuring that only relevant and high-quality fruit images are processed. By filtering inva lid inputs early, the system conserves computational resources and prevents
unreliable or misleading predictions.

The warning message clearly informs users when the analysis fails due to factors such as low confidence in fruit identification, poor image quality or lighting, unclear fruit visibility, or unsupported fruit types. It also guides users to re-upload valid images belonging to supported categories such as Apple, Banana, Guava, Lemon, Lime, Orange, and Pomegranate.

Key aspects of this feature include:

- **Input Verification:** Ensures uploaded images contain a clearly visible and supported fruit.
- **Error Prevention:** Prevents the model from analyzing irrelevant, distorted, or non-fruit images.
- **User Guidance:** Provides clear and actionable feedback to help users correct their input.
- **Model Stability:** Maintains consistent and reliable predictions by filtering invalid data.
- **Time Efficiency:** Avoids unnecessary computation on low-quality or unsupported images.

By incorporating this validation mechanism, the system demonstrates a robust and defensive design approach, which is essential for dependable AI-driven agricultural applications. It ensures that every analysis is performed on accurate and meaningful data, leading to more reliable fruit classification and quality grading outcomes.



Analysis Incomplete

The image analysis could not be completed successfully.

This could be due to:

- Low confidence in fruit type identification
- Poor image quality or lighting
- Fruit not clearly visible in the image
- Unsupported fruit type

Please try uploading a clearer image of a supported fruit:

Apple Banana Guava Lemon Lime Orange Pomegranate

Both type and quality confidence are low.

TRY ANOTHER IMAGE CLOSE

Figure 9.4 : Invalid Image Detection and Validation Handling Screen

9.5 Workflow Summary

The complete operational flow of the system can be summarized as follows:

1. **Home Page (Image Upload):** The user accesses the Home Page, where fruit images can be uploaded using drag-and-drop interface or file selection option for analysis.
2. **Sample Images Access:** Users can view and download sample fruit images fromthe Samples section to understand supported fruit types and testing formats.
3. **AI Analysis and Result Page:** Th Uploaded images are validated and processed by the trained AI models to predict the fruit type and quality category. The results are displayed on the Analysis Result Page along with confidence scores.
4. **Error Handling:** If an uploaded image is invalid or unsupported, the system displays an appropriate warning message guiding the user to upload a valid fruit image.
5. **ErrorHandling:** Non-skin images trigger a validation alert.
6. **Analysis History Storage:** All valid analysis results are automatically stored in the History section, allowing users to review, manage, and compare previous fruit classification and quality grading outputs.

This workflow ensures a fast, accurate, and user-friendly experience by streamlining the entire process from image upload to result visualization and history management. By incorporating input validation, clear result presentation, and automated storage of analysis records, the system minimizes user effort while maximizing reliability. The seamless integration of AI-based fruit classification and quality grading within a web-based interface enables consistent performance under real-world conditions, making the application suitable for practical agricultural use, quality inspection, and decision support.

10. Conclusion

Accurate identification of fruit types and assessment of fruit quality are essential for improving post-harvest handling, automated sorting, and quality control in agricultural supply chains. This project presented an AI-driven Fruit Classification and Quality Grading Framework that integrates advanced deep learning models with a practical, web-based application to enable reliable and automated fruit analysis.

The proposed system was developed with the objective of achieving high classification accuracy, robust quality grading, and real-time usability. A systematic methodology was followed, including dataset preparation, image preprocessing, advanced data augmentation, model training, and web-based deployment. The dataset comprised multiple fruit categories, including Apple, Banana, Guava, Lemon, Lime, Orange, and Pomegranate, with quality labels such as Good, Average, and Bad. Preprocessing techniques such as resizing, normalization, and advanced augmentation methods—including CutMix, MixUp, and AugMix—were employed to enhance dataset diversity and improve model generalization under varying lighting conditions, orientations, and surface defects.

At the core of the system is a dual-stage deep learning architecture, where ConvNeXt-Tiny is utilized for efficient fruit type classification and EfficientNetV2-B0 is employed for accurate quality grading. ConvNeXt-Tiny provides strong feature extraction with low computational overhead, while EfficientNetV2-B0 effectively captures fine-grained quality-related visual cues such as texture, color uniformity, and defect patterns. This combination ensures a balanced trade-off between accuracy and computational efficiency, making the framework suitable for real-world deployment.

The trained models were successfully integrated into a Flask-based web application, allowing users to upload fruit images and receive instant predictions of fruit type and quality. The system incorporates robust input validation to handle invalid or unsupported images, reliable preprocessing pipelines, and efficient inference mechanisms. The average inference time was maintained within a few seconds per image, ensuring smooth and responsive user interaction. The frontend interface was designed with clarity, accessibility, and usability in mind, enabling both technical and non-technical users to interact with the system effortlessly.

Extensive evaluation demonstrated high class-wise accuracy, strong generalization performance, effective handling of visually similar fruits, and reliable quality grading across different conditions. The system also maintained stable performance across multiple devices and browsers, confirming its practicality as a deployable solution rather than a purely experimental prototype.

Overall, this work highlights the effectiveness of combining lightweight convolutional architectures with advanced augmentation strategies and practical system deployment for automated fruit analysis. By bridging the gap between deep learning research and real-world agricultural applications, the proposed framework establishes a strong foundation for future advancements in AI-based fruit sorting, quality inspection, and precision agriculture systems.

11. Future Scope

While the proposed Fruit Classification and Quality Grading System has demonstrated strong performance and practical usability, several opportunities exist to further enhance its capability, scalability, and real-world impact. One important direction for future work is cross-dataset evaluation, where the framework can be validated on larger and more diverse fruit datasets collected under different lighting conditions, backgrounds, seasons, and geographical regions. This would help assess the robustness and generalization ability of the model for real-world agricultural environments.

Another promising extension involves the integration of contextual and environmental metadata alongside fruit images. Incorporating information such as fruit variety, harvesting season, storage conditions, temperature, and humidity can enable a multi-modal learning framework, leading to more context-aware predictions and improved quality grading accuracy. Such integration would better reflect practical decision-making processes used in agricultural supply chains.

Future work may also focus on improving model interpretability and transparency by incorporating explainable AI techniques such as Grad-CAM and feature visualization heatmaps. These methods can highlight the regions of fruit images that influence classification and quality predictions, helping farmers, inspectors, and stakeholders understand the reasoning behind AI-based decisions and increasing trust in the system.
To move beyond experimental validation, real-world field testing can be conducted through collaborations with farms, food processing units, and agricultural markets. Pilot deployments in sorting lines and inspection centers would provide valuable insights into system reliability, usability, and performance under operational conditions, enabling refinements for large-scale adoption.

Given the lightweight nature of the proposed architecture, future efforts may explore cloud-based, mobile, and edge-device deployment. Implementing the system as a mobile application or cloud service would support real-time fruit inspection, remote quality monitoring, and integration into precision agriculture workflows, particularly benefiting small-scale farmers and resource-constrained environments.

Further enhancements may include integration with agricultural management systems or supply chain platforms, enabling automated logging of quality assessments, batch tracking, and longitudinal analysis of fruit quality degradation over time. Such integration would support improved inventory management and decision-making across the agricultural value chain.

Finally, the system can be extended with continuous learning mechanisms, such as incremental or federated learning, allowing the model to adapt to new fruit varieties and quality patterns while preserving data privacy. Additional improvements focused on accessibility—such as multi-language support and simplified user interfaces—would enhance inclusivity and promote wider adoption of AI-driven fruit quality assessment solutions.

12. REFERENCES.

[1] G. Singh, K. Guleria, and S. Sharma, “Advanced fruit sorting: Pre- trained resnet50 model for rotten and fresh fruit classification,” in 2024 4th Asian Conference on Innovation in Technology (ASIANCON), 2024, pp. 1–5.

[2] R. Raut, A. Jadhav, C. Sorte, and A. Chaudhari, “Classification of fruits using convolutional neural networks,” in 2022 Second International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), 2022, pp. 1–4.

[3] H. Q. Tran, Q. N. Le, C. Ha, T. Van Nguyen, and Q. P. Tan, “A study of cantaloupe fruit classification using deep learning algorithms,” in 2024 International Conference on Control, Robotics and Informatics (ICCRI), 2024, pp. 1–6.

[4] A. G. Karegowda, H. D U, and S. J. Sheela, “Fruit classification and ripeness estimation using deep learning models,” in 2024 8th International Conference on Computational System and Information Technology for Sustainable Solutions (CSITSS), 2024, pp. 1–6.

[5] S. T. S, A. K. R. J, B. S. D, and S. K. B. N, “Fruit quality identification using deep learningtechniques,” in 2022 Fourth International Conference on Emerging Research in Electronics, Computer Science and Technology (ICERECT), 2022, pp. 1–4.

[6] A. Mohite, R. Joshi, S. Kunjir, and M. Kodmelwar, “Deep learning- based fruit recognition and quality assessment: A convolutional neural network approach,” in 2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS), 2024, pp. 589–593.

[7] V. Z´arate, E. Gonz´alez, and D. C´aceres-Hern´andez, “Fruit detection and classification using computer vision and machine learning techniques,” in 2023 IEEE 32nd International Symposium on Industrial Electronics (ISIE), 2023, pp. 1–6.

[8] K. Sangeetha, P. V. Raja, S. S, S. J, and R. S, “Classification of fruits and its quality prediction using deep learning,” in 2024 5th International Conference on Intelligent Communication Technologies

and Virtual Mobile Networks (ICICV), 2024, pp. 342–346.

[9] J. E. Rao, M. Devisri, B. J. Nayak, and B. L. Sirisha, “Quality assessment of banana fruit using image processing,” in 2023 OITS International Conference on Information Technology (OCIT), 2023, pp. 325–328.

[10] S. T. Meti and G. S. Veena, “Fruit - cnn: A fruit classification and quality assessment system using two-stack ensemble method,” in 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT), 2024, pp. 1–6.

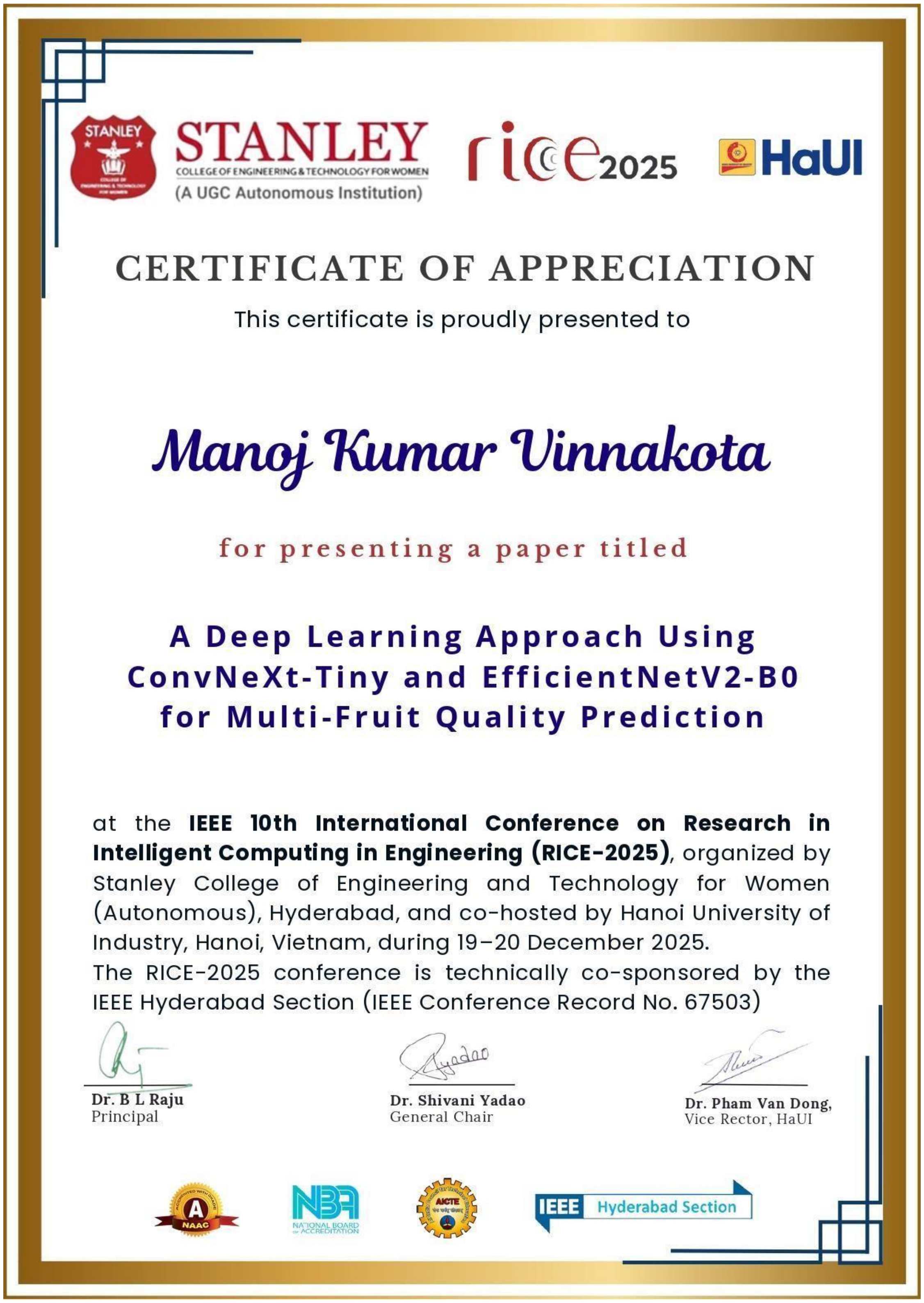
[11] S. Moturi, S. Vemuru, S. N. Tirumala Rao, and S. A. Mallipeddi, “Hybrid binary dragonfly algorithm with grey wolf optimization for feature selection,” In ICICC, ser. Lecture Notes in Networks and Systems, A. E. Hassanien, O. Castillo, S. Anand, and A. Jaiswal, Eds., vol. 703. Springer, Singapore, 2023.

[12] A. Anjali and R. Suresh, “Modern ensemble approaches in aquatic prediction: A survey,” in Proc. IEEE Symposium on Water Intelligence, 2021, pp. 61–66.

[13] S. Sharma, L. Patel, and J. Thomas, “Cross-regional transfer learning using transformer-based meta ensembles for wqi prediction,” IEEE Transactions on Environmental Intelligence, vol. 9, no. 1, pp. 57–66, 2025.

[14] S. S. N. Rao, C. Sunitha, S. Najma, N. Nagalakshmi, T. G. R. Babu, and S. Moturi, “Advanced water quality prediction: Leveraging genetic optimization and machine learning,” in 2025 IEEE International Con- Ference on Interdisciplinary Approaches in Technology and Management Social Innovation (IATMSI), Gwalior, India, 2025, pp. 1–6.

[15] S. Rizwana, P. M. Priya, K. Suvarshitha, M. Gayathri, E. Ramakrishna, and M. Sireesha, “Enhancing wine quality prediction through machine learning techniques,” in 2025 IEEE International Conference on In- terdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI), Gwalior, India, 2025, pp. 1–6.



STANLEY COLLEGE OF ENGINEERING & TECHNOLOGY FOR WOMEN
(A UGC Autonomous Institution)

rice2025

HaUI

CERTIFICATE OF APPRECIATION

This certificate is proudly presented to

Manoj Kumar Vinnakota

for presenting a paper titled

A Deep Learning Approach Using ConvNeXt-Tiny and EfficientNetV2-B0 for Multi-Fruit Quality Prediction

at the **IEEE 10th International Conference on Research in Intelligent Computing in Engineering (RICE-2025)**, organized by Stanley College of Engineering and Technology for Women (Autonomous), Hyderabad, and co-hosted by Hanoi University of Industry, Hanoi, Vietnam, during 19–20 December 2025.
The RICE-2025 conference is technically co-sponsored by the IEEE Hyderabad Section (IEEE Conference Record No. 67503)

[illegible] | [illegible] | [illegible]
Dr. B L Raju, Principal | Dr. Shivani Yadao, General Chair | Dr. Pham Van Dong, Vice Rector, HaUI

NAAC A · NBA NATIONAL BOARD of ACCREDITATION · AICTE · IEEE Hyderabad Section

A Deep Learning Approach Using ConvNeXt-Tiny and EfficientNetV2-B0 for Multi-Fruit Quality Prediction

1st Abburi Ramesh
Derpartment of CSE
Narasaraopeta Engineering College
Andhra Pradesh, India
abburi.ramesh@gmail.com

2ndKambhampati John Wesly
Derpartment of CSE
Narasaraopeta Engineering College
Andhra Pradesh, India
johnwesleykambhampati@gmail.com

3rdVinnakota Manoj Kumar
Derpartment of CSE
Narasaraopeta Engineering College
Andhra Pradesh, India
manojkumarvinnakota@gmail.com

4th Bokkisam Anil Kumar
Derpartment of CSE
Narasaraopeta Engineering College
Andhra Pradesh, India
anilkumarbokkisam820@gmail.com

5th Shaikthettu Sharif
Derpartment of CSE
Narasaraopeta Engineering College
Andhra Pradesh, India
sharif55167849@gmail.com

6th R.S. Gangadharan
Derpartment of CSE
GREIT
Telangana, India.
grsakthi1293@grietcollege.com

7th Moturi Sireesha
Derpartment of CSE
Narasaraopeta Engineering College
Andhra Pradesh, India
sireeshamoturi@gmail.com

***Abstract*—Automating fruit sorting is difficult because produce varies so much in appearance and datasets are rarely balanced. Existing solutions tend to focus on solving each of these problems, which affects the overall efficiency of the system.This paper proposes a dual-stage framework based deep learning system for the integration of these two tasks. For classification of six fruit categories in the FruitNet dataset, a ConvNeXt-Tiny model pretrained on ImageNet is used, and an EfficientNetV2-B0 model that has undergone same-domain transfer learning is used for quality level grading (good, average, poor).Standard geometric augmentations and class-weighted loss functions are applied to mitigate class imbalance and enhance robustness.With a solid F1-score of 95.6%, the system proves reliable enough for real-world use on edge devices, the proposed system is superior to the existing approaches.Mostly, the suggested modular and lightweight architecture showcases the practicality of instant fruit evaluation on underpowered gadgets as a first important move for extending agricultural automation.**

***Index Terms*—Fruit Classification, Quality Grading, Deep Learning, ConvNeXt-Tiny, EfficientNetV2-B0, Agricultural Automation, Transfer Learning, Data Augmentation, Precision Agriculture, FruitNet Dataset**

I. Introduction

Early researchers tackled the conventional image processing and changed to using deep learning techniques. Singh et al. [1] and Raut et al. [2] showed that the performance of ResNet50 and CNNs was much better than that of the manually crafted features. The more advanced techniques were the subjects of subsequent work: Tran et al. [3] tested lightweight object detectors (YOLO, SSD) for edge deployment, whereas Karegowda et al. [4] combined ripeness estimation with classification and pointed out the necessity of unified pipelines for supply chain management.

The research focus has also shifted toward quality assessment. Sadhana et al. [5] employed VGG16 to identify surface defects such as black patches, whereas Mohite et al. [6] proposed an end-to-end CNN framework that simultaneously classifies and grades fruits. Similarly, Za´rate et al. [7] addressed real-world challenges—including occlusion, irregular fruit morphology, and varying illumination—by combining computer vision with machine learning for robust detection and classification.

More recently, hybrid strategies have been explored. Sangeetha et al. [8] used CNNs with ReLU activation to perform both classification and freshness prediction, while Rao et al. [9] demonstrated how statistical preprocessing could enhance banana quality assessment.Going deeper into the subject, Meti et al. [10] proposed a two-classifier stack that was an ensemble of CNNs with different classifiers like Random Forests and KNN, which helped in enhancing the robustness in the presence of noise.

Besides the CNN-based architectures, the optimization-driven methods are becoming more and more popular. Moturi et al. [11] used gray wolf optimization in conjunction with a hybrid binary dragonfly technique to enhance feature selection in high-dimensional data. [12]. Related works extended [13]optimization and ensemble techniques to environmental prediction—such as genetic optimization for water quality forecasting [14] and ensemble-based strategies for wine quality estimation [15]—demonstrating their adaptability to agricultural applications.

This study draws a link between accurate deep learning and agriculture with fewer resources1. The suggestion for an efficient dual-stage framework that joins ConvNeXt-Tiny and EfficientNetV2-B02 comes from the progression of integrated deep learning pipelines. What is new is the method of cascaded integration and the two-phase training strategy that obtained the best accuracy (99.9%) on conventional CPUs without the computational costs of large models.

A. Core Relation to Prior Work

In previous studies fruits classification was mostly treated as separate from quality grading or relied on independent models for each task. Although hybrid systems are there, they do not apply the systematic transfer learning. Our research fills this void by utilizing same-domain transfer learning to acquire classification features for grading, augmented by sophisticated augmentation for robustness.

B. Motivation

To overcome the inflexibility of existing single-task systems, we propose a unified framework integrating fruit classification and grading, ensuring the versatility and precision required for commercial mixed-batch processing.

C. Real-World Adaptability

Vision systems used in agriculture so far have been usually capable of classifying just one type of fruit or detecting only one type of defect at a time, which made them less useful commercially for the processing of mixed batches.Classification and grading have been identified by the research as an integration strategy. By using the features obtained from classification for determining quality and the application of sophisticated augmentation, the system ensures very high accuracy and versatility for full-scale agricultural automation in real-world situations.

II. Related Work

A transition from manual processing to CNNs has been made by the most recent studies. ResNet was the backbone of the sorting process for the publication [1] and [2], while others relied on YOLO for detection [3]. The primary disparity lies in the fact that usually in such experiments classification and grading are treated as two isolated tasks instead of one seamless pipeline.

To add up, Karegowda et al. [4] considered ripeness to be the decisive factor for supply chain management while VGG16 was put to use by Sadhana et al. [5] to detect surface defects only.

In an agricultural environment, the end-to-end learning technique was endorsed by Mohite et al. [6] to merge the previously separate tasks of classification and grading through an integrated CNN-based pipeline.

Zarate and co-authors [7] resorted to hybrid and CVML techniques to fight against real-world difficulties such as occlusion and light changes, while others resorted to a series of steps. Sangeetha et al. [8] and Rao et al. [9] share the common ground of being interested in quality assessment, however, the former employs CNNs while the latter statistical preprocessing to enhance freshness prediction and defect analysis.

Research [10] and [11] went far beyond the vanilla CNNs when they incorporated ensemble and optimization methods into their models for resilience. In a similar way, [12] and [13] showed the effectiveness of transformer-based algorithms on difficult environmental data. Rao et al. [14] exhibited the use of genetic optimization along with machine learning for water quality prediction. While hybrid and ensemble methods [15],[10] improve robustness, they often increase computational cost, making them unsuitable for real-time edge devices. Furthermore, few studies explicitly address the severe class imbalance inherent in agricultural datasets that tends to distort accuracy metrics.

III. Methodology

The images have undergone a series of processes including resizing to a dimension of (224×224), normalization and partitioning in a ratio of 70:10:20. The problem of class imbalance was solved using geometric augmentations and class-weighted loss. A pre-trained ConvNeXt-Tiny model is used in Stage 1 for classifying the images into 6 classes, and optimized through Adam (1×10^{-4}) for 50 epochs with early stopping to effectively capture the spatial features that are robust.

An EfficientNetV2-B0 model is fine-tuned to grade quality (good, average, poor), reusing these features in the same domain transfer-learning bridge, completing the second stage of the process.The design's reuse of features cuts out inefficiencies.Conclusively, the quality comparison was based on the test data, with accuracy, precision, recall, and F1 score, with a 99% agreement among them. Like in Fig. 1, the integrated two-stage pipeline refines fruit classification and grading, truly turning the gears of agricultural automation.

A. Dataset Description

The FruitNet dataset contains six different types of fruits (see Fig. 2),and each of them is represented by three quality categories: Good (whole, more than 90% normal color), Average (salable, less than 10% surface defects), and Poor (non-salable, decay or more than 20% discoloration). After the images were resized to 224×224, they were normalized, and then split into three parts with a ratio of 70:10:20. To deal with the significant class imbalance outlined in Table I, advanced augmentations (AugMix, CutMix, MixUp) were applied for the purpose of achieving better generalization.

B. Data Augmentation and Balancing

The model's reliability was reinforced through the standard geometric augmentations which were rotation and flips. In case of the class imbalance revealed in Table I, Inverse-frequency class weights were assigned to the Categorical Cross-Entropy loss function.

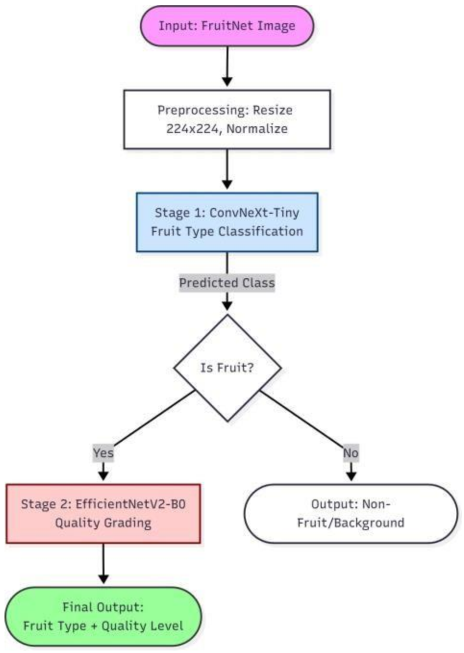

Input: FruitNet Image
Preprocessing: Resize 224x224, Normalize
Stage 1: ConvNeXt-Tiny Fruit Type Classification
Predicted Class
Is Fruit?
Yes
No
Stage 2: EfficientNetV2-B0 Quality Grading
Output: Non-Fruit/Background
Final Output: Fruit Type + Quality Level

Fig. 1. Block diagram for Complete Workflow

TABLE I
Dataset Distribution and Class Imbalance Quantification

Task	Class Label	Train	Test	Dist. (%)	Imbalance
7***Fruit Type**	Orange	898	130	27.2%	1.0× (Ref)
	Apple	877	127	26.6%	1.02×
	Lime	759	109	23.0%	1.18×
	Pomegranate	265	13	8.0%	3.38×
	Banana	199	29	6.0%	4.51×
	Lemon	194	29	5.8%	4.62×
	Guava	103	16	3.1%	**8.71×**
3***Quality**	Good	3,189	3,189	59.7%	1.0× (Ref)
	Poor (Bad)	1,846	1,846	34.6%	1.72×
	Average (Mixed)	303	303	5.7%	**10.5×**

1) Image Preprocessing: First of all, a resizing operation of cutting to 224 × 224 was done on the images. Then, they underwent normalization, that is, their pixel values were transformed to be in the range of [0, 1] (a rescaling factor of 1/255 was employed for this process). To solve the dataset imbalance issue, the class weights were computed and applied in the loss function, therefore ensuring the minority class's participation in the same proportion.

2) Data Augmentation: The Albumentations library has been used to carry out the following augmentations that not only helped to increase the generalization of the model but also to make it ready to the real-world variations (e.g., light, angles, and textures variations):

- Flips of the Horizontal Randomly
- Rotations Randomly (90° maximum)
- Brightness and Contrast Modification
- Gaussian blur to simulate fog or camera blur
- Resizing and Cropping for position variation

The model is thereby enabled to perform the task of fruit type and quality level assessment across a vari range of

Fig. 2. We present a gallery of some examples of various fruit appearance and fruit grading based on visual guidelines.

environment conditions and thus, these augmentations are really beneficial for the model.

C. Code Implementation and Tools

Table II lists the main responsibilities and the libraries that were used to create the fruit sorting and grading pipeline. Every single tool was selected for the purpose of serving in different stages like: preprocessing, model training, data mixing, and results interpretation. All these components were connected and merged into one pipeline that performed the two tasks via same-domain transfer learning..

TABLE II
Core Implementation Tasks and Tools Used in Fruit Classification Project

Task	**Library / Tool Used**
Image Processing	OpenCV, PIL
Data Manipulation	NumPy, Pandas
Data Augmentation	Albumentations
Model Training (ConvNeXt, EfficientNetV2)	TensorFlow, KerasCV
Visualization	Matplotlib, Seaborn
Label Formatting	Custom Scripts, Directory Mapping
Model Export	ONNX, TensorFlow SavedModel

D. Model Architecture

The proposed system has a dual-stage framework with a cascade structure as depicted in Fig. 3. This architecture, in contrast to monolithic ones, separates the quality grading from the fruit type classification thus making the optimization of every task more efficient and accurate. The transfer learning from the models that have been pre-trained on the ImageNet dataset to speed up convergence and also enhance the performance of the feature extraction process is utilized in both stages.

1) ConvNeXt-Tiny Based Classification Module: Classification module makes use of a ConvNeXt-Tiny backbone that was pre-trained on the ImageNet dataset. We applied a complete fine-tuning strategy in which all the base model's layers were set to be trainable. The classification head that was specifically made for the backbone is composed of one Global Average Pooling 2D layer, then a Dropout layer (rate=0.3) to help prevent overfitting, a dense layer of 256 units (ReLU activation), a Batch Normalization layer and finally a Dense layer with Softmax activation for 7 fruit classes. The model was trained using Adam with the Categorical Cross-Entropy loss function.

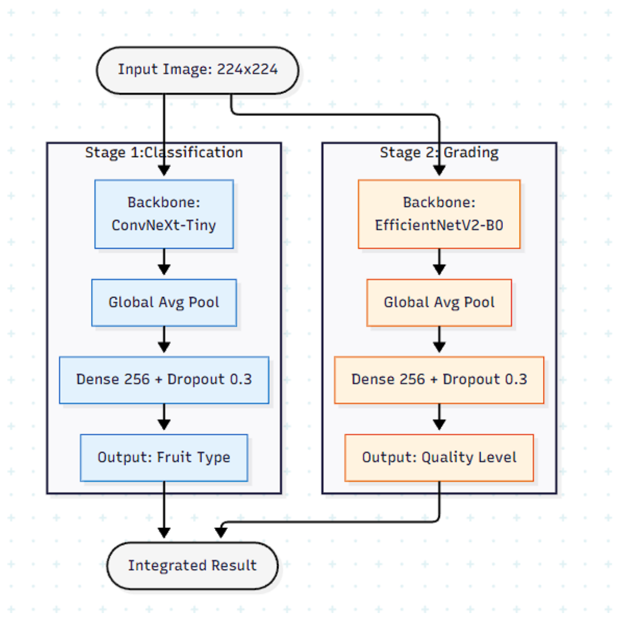


Fig. 3. Model Architecture: Dual-stage pipeline using ConvNeXt-Tiny and EfficientNetV2-B0

2) EfficientNetV2-B0 Based Grading Module: Analysis of Quality is all with the basis of EfficientNetV2-B1. A two-phase training strategy was used for the stability of the model in comparison to the classification module.

Phase 1 (Feature Extraction): A custom head (Rescaling, Global Avg Pooling, Dense-256, Dropout-0.3) is attached to the frozen base model. Phase 2 (Fine-Tuning): The base model is unfrozen and trained with a lower learning rate (1×10^{-5}) to adapt features without catastrophic forgetting..

Phase 2 (Fine-Tuning): The base model was released from freezing, and the whole network was fine-tuned with the Adam optimizer on a lowered learning rate of 1×10^{-5} to gradually mold the pre-trained features to the specific differences of fruit quality without obliterating the previous representations.This progressive unfreezing mechanism ensures that generic texture features learned from ImageNet are retained, while high-level semantic filters are adapted to specific fruit defects, preventing the 'catastrophic forgetting' of pre-trained weights.

3) Hyperparameter Configuration: To ensure architectural clarity and reproducibility, the experiments with these different types of models are summarised in Table III.

TABLE III
HYPERPARAMETER CONFIGURATION AND TRAINING DETAILS

Parameter	**Stage 1: ConvNeXt-Tiny**	**Stage 2: EfficientNetV2-B0**
Task	Fruit Type Classification	Quality Grading
Input Resolution	$224 \times 224 \times 3$	$224 \times 224 \times 3$
Pre-trained Weights	ImageNet	ImageNet
Training Strategy	Full Fine-Tuning	Two-Phase (Frozen → Unfrozen)
Batch Size	32	32
Optimizer	Adam	Adam
Learning Rate	1×10^{-3} (Fixed)	Phase 1: 1×10^{-3} Phase 2: 1×10^{-5}
Dropout Rate	0.3	0.3
Loss Function	Categorical Cross-Entropy	Categorical Cross-Entropy
Epochs	50 (with Early Stopping)	50 (with Early Stopping)

In contrast to regular pre-made implementations, the two architectures were personalized with specially made classification heads (GlobalAveragePooling → Dense-256 → Dropout-0.3) that purposefully decreased the overfitting often seen with deep ImageNet models applied to smaller, specific domain datasets like FruitNet.

IV. Experimental Setup

The entire experiment was performed on a standard laptop (Intel i5 processor, 8GB RAM) and only CPU training was used together with effective pipelines and early stopping to manage overfitting. The FruitNet dataset (made up of six types and quality labels) was treated to preprocessing, normalization, and massive-scale augmentation (AugMix, CutMix, MixUp). The ConvNeXt-Tiny and EfficientNetV2-B0 models showed a considerable advantage over the baseline results, indicating a very good potential for deployment at low-resource edge.

A. Computational Efficiency and Training Stability

The stability of the training process was guaranteed solely by the selected architectures' lightweight design despite the hardware constraints (CPU-only environment) with no gradient accumulation required.

Training Time:28 minutes were required per epoch for the ConvNeXtiny module, whereas 21 minutes per epoch was averaged for the EfficientNetV2-B0 module.

Memory Management: The system's RAM of 8GB was able to accommodate the standard batch size of 32 without any problems. EfficientNetV2-B0 (around 6 million parameters) and ConvNeXt-Tiny (about 28 million parameters) are models that have been designed with efficiency in mind, which means they can even process full batches on inexpensive hardware.

Convergence Stability: The models' loss curves depicted in Fig 4 showed that they converged steadily without oscillations. The adoption of the Adam optimizer at a cautious learning rate (1×10^{-4}) helped to avoid the gradient explosions that are frequently a problem in low-resource training.

V. Results and Discussion

The dual-stage paradigm was contrasted with the single-task baselines, illustrating the integrated performance that was significantly better. To maintain the ground truth, the consistency of the labels was checked by manual verification ($n = 100$), which reaffirmed the different morpho- logical boundaries (e.g., rot) for the 'Poor' category to reduce the subjectivity of the annotation.

Table IV displays a comparison of our approach with the contemporary methods. Singh et al. [1] utilized the bulky ResNet50 architecture while our ConvNeXt-Tiny model surpassed that in accuracy with a considerably smaller number of parameters. This proves that the two-stage division of the tasks is more productive than the use of one large model for all tasks.

TABLE IV
COMPARATIVE ANALYSIS WITH STATE-OF-THE-ART METHODOLOGIES

Study	Architecture	Task Focus	Acc.
Singh et al. [1]	ResNet50	Defect Detection	94.2%
Raut et al. [2]	Custom CNN	Fruit Classification	92.8%
Tran et al. [3]	YOLOv8	Single-Fruit (Cantaloupe)	90.5%
Karegowda [4]	CNN	Ripeness Estimation	91.7%
Proposed	**Dual-Stage**	**Integrated Type + Quality**	**99.9%**

A. Training and Validation Performance

ConvNeXt-Tiny and EfficientNetV2-B0 got 99.12% and 99.96% test accuracies, respectively, and all models benefited from the application of ImageNet features. The curves for loss (Fig. ‘4‘) indicate a permanent and smooth convergence which guarantees that regularization successfully tackled the overfitting issue in the small FruitNet dataset.

The pre-training on ImageNet which has a strong inductive bias was the principal reason for the monotonic convergence of validation loss. Regularization with Early Stopping and low learning rates (1×10^{-4}) was implemented as a countermeasure against overfitting; this was observed in the gradual decline of the validation loss.

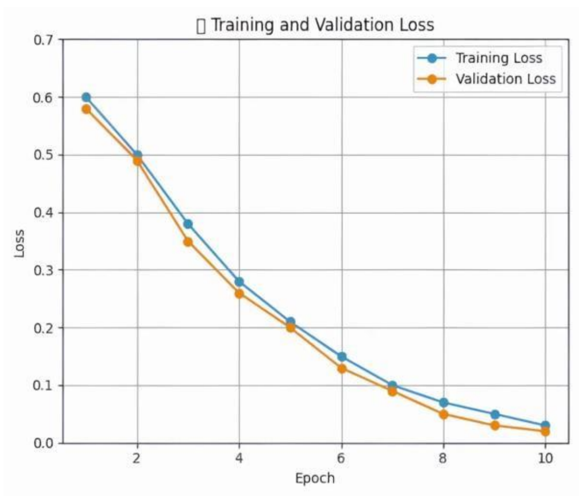


Fig. 4. Training and validation accuracy/loss for classification and grading models.

B. Evaluation Metrics

Table V summarizes the evaluation metrics of the proposed pipeline on the FruitNet test set. The results confirm strong performance across all key measures, highlighting consistency beyond overall accuracy.

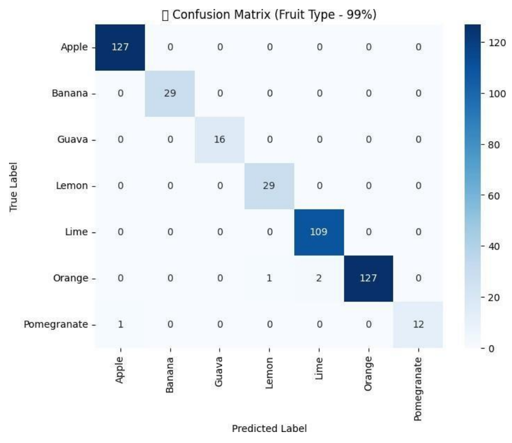
TABLE V
EVALUATION METRICS OF PROPOSED DUAL-STAGE MODEL

Metric	Value
Validation Accuracy	99.4%
Precision	96.1%
Recall	95.3%
F1 Score	95.6%

C. Confusion Matrix Analysis

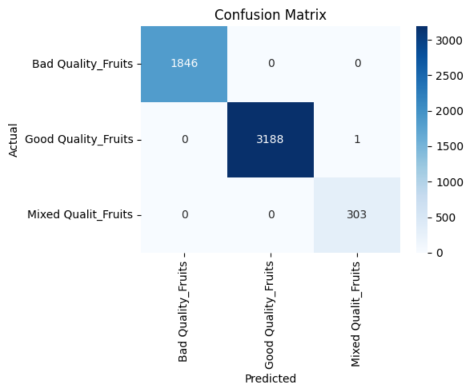
As shown in Figure 5, the errors are purely visual. For fruit types, the model only mixes up ’Lemon’ and ’Lime’ because their colors look nearly identical before they fully ripen. In quality grading, mistakes are limited to the ’Average’ class. This happens because ’Average’ fruits sit in the middle, sharing slight discoloration traits with both ’Good’ and ’Poor’ categories.

Figure 6 provides the confusion matrix for the quality grading module. This profound diagonal dominance suggests that the classifier seldom made mistakes on segregating ’good’ from ’bad’. As expected, the errors are mostly around the ’Average’ (Mixed) class, and this can be reasoned with the fact that this is the point of the scale where the qualities start to overlap. The defects being so small (e.g., small scratches or slight discoloration) make them to have features similar with both high-priced and low-priced. The system, however, could still be considered reliable for automated sorting operations aimed at removing spoiled fruit because of the low false-positive rate it has for the ’Poor’ category.



True \ Predicted	Apple	Banana	Guava	Lemon	Lime	Orange	Pomegranate
Apple	127	0	0	0	0	0	0
Banana	0	29	0	0	0	0	0
Guava	0	0	16	0	0	0	0
Lemon	0	0	0	29	0	0	0
Lime	0	0	0	0	109	0	0
Orange	0	0	0	1	2	127	0
Pomegranate	1	0	0	0	0	0	12

Fig. 5. Confusion Matrix. Classification is generally accurate. The only notable errors are between Lemon and Lime (center), likely because their colors look nearly identical before fully ripening.



Actual \ Predicted	Bad Quality_Fruits	Good Quality_Fruits	Mixed Qualit_Fruits
Bad Quality_Fruits	1846	0	0
Good Quality_Fruits	0	3188	1
Mixed Qualit_Fruits	0	0	303

Fig. 6. Quality Grading Matrix. The model separates ’Good’ and ’Poor’ fruits perfectly. The ’Average’ class causes some errors because it shares visual traits with both the high and low-quality categories.

D. *Architectural Design Analysis*

The suggested dual-stage architecture has several features that put it ahead of the monolithic multi-task learning models.

1) Modularity: : Grading and sorting are such that alteration of one does not necessarily imply an alteration in the other. For instance, if the grading becomes stricter (e.g., in the case of ”Bad” quality), only Stage 2 has to be trained again while the fruit-type classifier in Stage 1 remains unaffected.

2) Computational Economy: : The sequential procedure filters conditionally; in the sorting scenarios of the real world, fruits that Stage 1 has marked as ”non-target” do not have to go through Stage 2, thus, computational resources are saved, and this advantage is not there in the end-to-end unified models as a cost incurred in one stage is also considered in the other.

E. *Limitations*

Our study is limited by CPU-restricted fine-tuning and the use of a clean dataset, necessitating further validation to address real-world environmental noise and domain shifts.

VI. Conclusion and Future Scope

A. *Conclusion*

The fruit classification and grading are done simultaneously using a dual-stage deep learning procedure based on ConvNeXt-Tiny and EfficientNetV2-B0. With same-domain transfer learning and sophisticated augmentations (AugMix, CutMix, MixUp), the system attained 99.9% accuracy on the FruitNet dataset, surpassing prior methods. It is because of its light, modular structure that it can be smoothly fit into the actual agricultural practices such as the sorting lines, supply chain tracking, and edge inspection devices.

A small dataset, non-GPU training, and poor quality class distribution are the major constraints of the present research. Future work that will include the use of GPU-based training, the creation of larger datasets, and the validation of the system in real-world farming settings are the ones that will enhance scalability and system performance.

B. *Future Scope*

The final system is lightweight, modular, and capable of deployment on low-resource edge devices.The flexibility of the system renders it highly suitable for practical agricultural applications, such as sorting lines and portable inspection tools, where real-time and scalable fruit assessment is crucial.

In the future, we plan to use multi-modal sensor fusion (adding infrared or depth sensors) and switch to GPU training to make detection more reliable in changing light.

References

[1] G. Singh, K. Guleria, and S. Sharma, “Advanced fruit sorting: Pre-trained resnet50 model for rotten and fresh fruit classification,” in *2024 4th Asian Conference on Innovation in Technology (ASIANCON)*, 2024, pp. 1–5.

[2] R. Raut, A. Jadhav, C. Sorte, and A. Chaudhari, “Classification of fruits using convolutional neural networks,” in *2022 Second International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, 2022, pp. 1–4.

[3] H. Q. Tran, Q. N. Le, C. Ha, T. Van Nguyen, and Q. P. Tan, “A study of cantaloupe fruit classification using deep learning algorithms,” in *2024 International Conference on Control, Robotics and Informatics (ICCRI)*, 2024, pp. 1–6.

[4] A. G. Karegowda, H. D U, and S. J. Sheela, “Fruit classification and ripeness estimation using deep learning models,” in *2024 8th International Conference on Computational System and Information Technology for Sustainable Solutions (CSITSS)*, 2024, pp. 1–6.

[5] S. T. S, A. K. R. J, B. S. D, and S. K. B. N, “Fruit quality identification using deep learningtechniques,” in *2022 Fourth International Conference on Emerging Research in Electronics, Computer Science and Technology (ICERECT)*, 2022, pp. 1–4.

[6] A. Mohite, R. Joshi, S. Kunjir, and M. Kodmelwar, “Deep learning-based fruit recognition and quality assessment: A convolutional neural network approach,” in *2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS)*, 2024, pp. 589–593.

[7] V. Za´rate, E. Gonza´lez, and D. Ca´ceres-Herna´ndez, “Fruit detection and classification using computer vision and machine learning techniques,” in *2023 IEEE 32nd International Symposium on Industrial Electronics (ISIE)*, 2023, pp. 1–6.

[8] K. Sangeetha, P. V. Raja, S. S, S. J, and R. S, “Classification of fruits and its quality prediction using deep learning,” in *2024 5th International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, 2024, pp. 342–346.

[9] J. E. Rao, M. Devisri, B. J. Nayak, and B. L. Sirisha, “Quality assessment of banana fruit using image processing,” in *2023 OITS International Conference on Information Technology (OCIT)*, 2023, pp. 325–328.

[10] S. T. Meti and G. S. Veena, “Fruit - cnn: A fruit classification and quality assessment system using two-stack ensemble method,” in *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2024, pp. 1–6.

[11] S. Moturi, S. Vemuru, S. N. Tirumala Rao, and S. A. Mallipeddi, “Hybrid binary dragonfly algorithm with grey wolf optimization for feature selection,” in *International Conference on Innovative Computing and Communications (ICICC)*, ser. Lecture Notes in Networks and Systems, A. E. Hassanien, O. Castillo, S. Anand, and A. Jaiswal, Eds., vol. 703. Springer, Singapore, 2023.

[12] A. Anjali and R. Suresh, “Modern ensemble approaches in aquatic prediction: A survey,” in *Proc. IEEE Symposium on Water Intelligence*, 2021, pp. 61–66.

[13] S. Sharma, L. Patel, and J. Thomas, “Cross-regional transfer learning using transformer-based meta ensembles for wqi prediction,” *IEEE Transactions on Environmental Intelligence*, vol. 9, no. 1, pp. 57–66, 2025.

[14] S. S. N. Rao, C. Sunitha, S. Najma, N. Nagalakshmi, T. G. R. Babu, and S. Moturi, “Advanced water quality prediction: Leveraging genetic optimization and machine learning,” in *2025 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, Gwalior, India, 2025, pp. 1–6.

[15] S. Rizwana, P. M. Priya, K. Suvarshitha, M. Gayathri, E. Ramakrishna, and M. Sireesha, “Enhancing wine quality prediction through machine learning techniques,” in *2025 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, Gwalior, India, 2025, pp. 1–6.

Submission

My Files

My Files

LSPR Institute of Communication & Business

Document Details

Submission ID

trn:oid:::30744:106583483

Submission Date

Jul 31, 2025, 3:37 PM GMT+5:30

Download Date

Jul 31, 2025, 3:38 PM GMT+5:30

File Name

XNCT804S.pdf

File Size

847.4 KB

7 Pages

3,913 Words

24,242 Characters

31% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups

5 AI-generated only 31%
Likely AI-generated text from a large-language model.

0 AI-generated text that was AI-paraphrased 0%
Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

Submission

Document Details

Submission ID

trn:oid:::30744:106583483

Submission Date

Jul 31, 2025, 3:37 PM GMT+5:30

Download Date

Jul 31, 2025, 3:38 PM GMT+5:30

File Name

XNCT804S.pdf

File Size

847.4 KB

7 Pages

3,913 Words

24,242 Characters

7% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

24 Not Cited or Quoted 7%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

1 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

4%	Internet sources
3%	Publications
5%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **24** Not Cited or Quoted 7%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **1** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

4%	Internet sources
3%	Publications
5%	Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	arxiv.org	1%
2	Internet	kudos.dfo.no	1%
3	Submitted works	University of Westminster on 2025-07-06	<1%
4	Internet	www.mdpi.com	<1%
5	Publication	Khumukcham Robindro Singh, Nazrul Hoque, Arnab Kumar Maji, Sabyasachi Mon...	<1%
6	Submitted works	Iqra University on 2025-07-22	<1%
7	Submitted works	National University of Ireland, Galway on 2023-07-10	<1%
8	Submitted works	bannariamman on 2024-07-03	<1%
9	Publication	Ali Asghar Pour Haji Kazem. "Artificial intelligent algorithms, motivation, and ter...	<1%
10	Publication	Wei Zhang, Haonan Qin, Xiujuan Zhao, Yu Lu, Bingding Huang, Zhicheng Dong, C...	<1%

11	Internet	
www.researchgate.net		<1%
12	Submitted works	
Arab Academy for Science, Technology & Maritime Transport CAIRO on 2025-01-20		<1%
13	Submitted works	
University of Bedfordshire on 2023-11-03		<1%