DeepFruit: A Unified Deep Learning Framework for Fruit Classification and Quality Evaluation

A Project Report submitted in the partial fulfillment

of the Requirements for the award of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Bollineni Yasaswini (21471A0512)

Aravapalli Vasavi (21471A0504)

Nakka Annapurna (21471A0538)

Under the esteemed guidance of

Dr. S.N.Tirumala Rao, M.Tech , Ph.D.

Professor & HOD

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NARASARAOPETA ENGINEERING COLLEGE: NARASARAOPET

(AUTONOMOUS)

Accredited by NAAC with A+ Grade and NBA under Tier -1
NIRF rank in the band of 201-300 and an ISO 9001:2015 Certified

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

KOTAPPAKONDA ROAD, YALLAMANDA VILLAGE,
NARASARAOPET- 522601

2024-2025

NARASARAOPETA ENGINEERING COLLEGE

(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project that is entitled with the name "DeepFruit: A Unified Deep Learning Framework for Fruit Classification and Quality Evaluation" is a bonafide work done by the team **Bollineni Yasaswini (21471A0512), Aravapalli Vasavi (21471A0504), Nakka Annapurna (21471A0538)** in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in the Department of **COMPUTER SCIENCE AND ENGINEERING** during 2024-2025.

PROJECT GUIDE	PROJECT CO-ORDINATOR
Dr. S.N.Tirumala Rao, M. Tech., Ph.D.	Dodda Venkata Reddy, M.Tech.,(Ph.D.)
Professor & HOD	**Assistant Professor**

HEAD OF THE DEPARTMENT	EXTERNAL EXAMINER
Dr. S. N. Tirumala Rao, M.Tech., Ph.D.	
Professor & HOD	

DECLARATION

We declare that this project work titled "**DEEPFRUIT: A UNIFIED DEEP LEARNING FRAMEWORK FOR FRUIT CLASSIFICATION AND QUALITY EVALUATION**" 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 submitted for any other degree or professional qualification except as specified.

Bollineni Yasaswini (21471A0512)
Aravapalli Vasavi (21471A0504)
Nakka Annapurna (21471A0538)

ACKNOWLEDGEMENT

We wish to express our thanks to various 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 this course. We owe out sincere gratitude to our beloved principal **Dr. S. Venkateswarlu**, **M. Tech., 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 **Dr. S. N. Tirumala Rao, M.Tech.,Ph.D.,** HOD of CSE department whose valuable guidance and unstinting encouragement enable us to accomplish our project successfully in time.

We extend our sincere thanks towards **Dodda Venkata Reddy**, **M.Tech.,(Ph.D.),** Assistant professor & Coordinator of the project for extending his encouragement. Their 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 our doubts which had really helped us in successfully completing our project.

By

Bollineni Yasaswini (21471A0512)
Aravapalli Vasavi (21471A0504)
Nakka Annapurna (21471A0538)

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 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.

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.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Outcomes

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activitieswith an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engagein independent and life-long learning in the broadest context of technological change.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Project Course Outcomes (CO'S):

CO421.1: Analyse the System of Examinations and identify the problem.

CO421.2: Identify and classify the 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: Prepare the 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	PO12	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	PO12	PSO1	PSO2	PSO3
C421.1	2	3											2		
C421.2			2		3								2		
C421.3				2		2	3	3					2		
C421.4			2			1	1	2					3	2	
C421.5					3	3	3	2	3	2	2	1	3	2	1
C421.6									3	2	1		2	3	

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

1. Low level
2. Medium level
3. High level

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

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

Name of the course from which principles are applied in this project	Description of the Task	Attained PO
C2204.2, C22L3.2	Gathering the requirements and defining the problem, plan to develop a model for simultaneously Classifying and Grading the fruits using Deep Learning models	PO1, PO3
CC421.1, C2204.3, C22L3.2	Each and every requirement is critically analyzed, the process model is identified and divided into 5 modules preprocessing, splitting, training, evaluating and predicting	PO2, PO3
CC421.2, C2204.2, C22L3.3	Logical design is done by using the unified modelling language which involves individual team work	PO3, PO5, PO9
CC421.3, C2204.3, C22L3.2	Each and every module is tested, integrated, and evaluated in our project	PO1, PO5
CC421.4, C2204.4, C22L3.2	Documentation is done by all our four members in the form of a group	PO10
CC421.5, C2204.2, C22L3.3	Each and every phase of the work in group is presented periodically	PO10, PO11
C2202.2, C2203.3, C1206.3, C3204.3, C4110.2	Implementation is done and the project will be leveraged by agricultural e-commerce platforms to verify the quality and type of produce sold online.	PO4, PO7
C32SC4.3	Designing a web interface to visualize predictions and verify model accuracy effectively	PO5, PO6

ABSRTACT

The simultaneous categorization and rating of fruit is an important but underexplored factor of personal computer (pc) vision in agricultural automation. Most previous studies conducted in this area have focused on capabilities consisting of size, shape, and shade, or utilized CNNs for the fruit category. The current study probes into the effectiveness of pre-trained deep learning fashions for fruit classification and quality assessment. In order to enable the transfer learning-based approach for multi-fruit type and grading, the approach mainly uses architectures such as MobileNetV2, EfficientNetV2, NASNet, DenseNet, VGG16, InceptionV3, and Xception. Our dual-model framework has been using those architectures for the classification of six high-quality fruit varieties, such as banana, apple, orange, pomegranate, lime, and guava, from the FruitNet Indian Fruits dataset containing 19,400 snapshots. These extracted weights are then fine-tuned to a 2D model that specializes in high-class grading. It has been observed that the application of data preprocessing techniques significantly improved the average overall performance of the model. Amongst the models studied, the MobileNetV2 completed the best test accuracy of 99.7% and hence emerged to be the best.

Other functionalities showed overall performance in the following manner: EfficientNetV2 99.5%, DenseNet 98%, NASNet 97.6%, Xception 97%, VGG16 96%, and InceptionV3-based totally models 95.75%. The present study reviews using transfer learning with one of the advanced deep learning models, namely MobileNetV2, to improve fruit type and grading for agricultural automation. By leveraging data pre-processing strategies, MobileNetV2 could portray state-of-the-art performance with the highest accuracy of 99.7%. It gives this approach a good-sized capacity to reinforce machine prescient structures within agriculture. As fine-grained visible analytics expand, this is able to deal with broader cultivation demanding situations. The investigation shows great potential for the development of this approach into device vision systems to reinforce agricultural automation. As fine-grained visible analytic capabilities continue to expand, this approach should be expanded to handle more general challenging cultivation

INDEX

S.NO.	LIST OF FIGURES	PAGE NO

1. INTRODUCTION

1.1 Introduction

Fruits deliver a wealth of nutrients and minerals, making them essential for human vitamins [1]. Fruit quality is historically decided often through visible inspection so that it will classify the type, freshness, and fine of the fruit. These critiques, finished by means of skilled group of workers, entail assessing numerous traits, consisting of coloration, shape, length, ripeness, and cleanliness, to make sure that culmination fulfill market requirements [2]. However, as generation has superior, this hobby has changed notably. The agriculture enterprise advanced to emerge as computer imaginative and prescient due to the fact to the exertions-intensive and exertions-extensive nature of manual inspection as well as growing societal requirements and expectancies [3]. The creation of device imaginative and prescient structures in agriculture, in particular for automating duties together with fruit sorting and grading, has progressed the manufacturing speed, overall performance, and accuracy of first-rate assessment at the same time as simultaneously lowering regular manufacturing prices[4][5]. These systems were broadly followed inside the agricultural organization for several responsibilities in pre-harvesting, harvesting, and post-harvesting end result and vegetables [6].

Fig 1.1 Importance of Fruits

At their center, these automation structures make use of tool getting to know algorithms that digitally interpret snap shots to determine fruit traits and extremely good via amassed revel in, similar to human notion [7]. Deep learning(DL), a complicated

subset of machine analyzing, now stands at the vanguard of strategies applied for fruit type and remarkable evaluation, amongst extraordinary related obligations[8]. In present day years, convolutional neural network (CNN) strategies, as one of the most well-known image elegance DL architectures, were substantially researched for fruit class and brilliant grading. Nevertheless, plenty of the studies over the last decade [9] has been constrained to studying character fruit sorts, which constrains the practicality of such strategies in numerous agricultural and food company settings wherein multiple fruit kinds are processed concurrently [10]. To deal with this hole, our examine introduces a versatile approach capable of classifying and grading a couple of fruit kinds. We employed a transfer reading methodology principally grounded in the multi-tiered MobileNetV2 [11] shape.

To classify the diverse final product, we first knowledgeable a model. Then, the usage of the taught weights from the primary level to grade fruit nice, we delicate all next variations via using transfer studying [12]. With the purpose of improving the scalability and use of machine imaginative and prescient in agriculture, this diploma technique offers a mixed answer for the automated categorization and grading of a few fruit varieties [13].

Fruits are a cornerstone of a healthy diet, offering a rich source of vitamins, minerals, and antioxidants vital for human nutrition [14]. Historically, assessing the quality of fruits has relied heavily on manual visual inspection to determine attributes such as type, freshness, and overall quality. This process involves skilled personnel evaluating various characteristics like color, shape, size, ripeness, texture, and cleanliness to ensure the fruits meet market standards [15]. While effective in small-scale operations, such manual methods are time-consuming, labor-intensive, and prone to human error, limiting their scalability and consistency [16].

1.2 Motivation

1) Simultaneous Multi-Fruit Classification and Grading : This study addresses the often disregarded mission of concurrently dealing with each class and grading obligations for more than one fruit kinds, in contrast to preceding studies that has typically centered on one assignment without thinking about the other [8][11].

2) This observe hypothesizes the powerful use of equal-area transfer studying by training a CNN model to classify a couple of culmination after which using the discovered expertiseto tell a subsequent grading version based on quality [9] .

3) This sequential twin-mission method is anticipated to streamline the classification and grading of culmination, improving the accuracy and performance of those procedures [12][16].

1.3 Contribution

1) An current application of identical-domain transfer getting to know is added, making use of a DL technique to transfer know-how from fruit kind class to fruit first-rate assessment [9][13].

2) The approach checks the performance of the MobilenetV2 DL structure, emphasizing training pace and reduced computational prices [11].

3) It represents the overall performance that the FruitNet dataset gives regarding the responsibilities of the fruit classes [6][15].

4) Advanced statistics pre-processing techniques, in particular, image Loading and Resizing, Label Mapping and Encoding, Handling Different Qualities Categories(Good, Bad, Mixed), coloration conversion, records Normalization are carried to decorate the version's robustness and generalization in classifying fruit types and the identification of fruit high-quality [7][8].

5) Records are split into the ratio of 80% for training and 20% for testing [10].

Fig1.3.1Good_Quality_Fruits

Fig1.3.2 Bad_Quality_Fruits

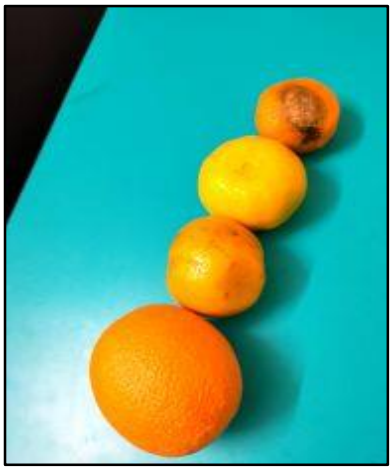

Fig1.3.3 Mixed_Quality_Fruits

2. LITERATURE SURVEY

In the 2024 exam, the emphasis is on how EfficientNetV2 applies to Same-Domain Transfer Learning during multi-fruit categorization. The proposed technique, using the FruitNet dataset, showed exquisite accuracy by the researchers. The examination shows how rightly EfficientNetV2 can classify distinct forms of culminations; extra studies are suggested to become privy with multi-challenge mastery for coping with tasks of a more complex type [1].

In 2024, another research into the progress of sorting fruit technology used public and private databases in combination. During examination, the AFC-HPODTL model has been presented, and it had an excellent performance in sorting. This study makes use of standard image information approaches; still, the authors gave suggestions for further refinement of styles so as to have higher precision in further research [2].

The paper proposes a smart fruit wonderful classifier using the FruitsGB dataset in the year 2024. Various deep mastering fashions with transfer studying strategies were used to avail extremely excellent type outcomes. Thus, the researchers underlined their technique as effective and called for further studies that would give more use to modern-day designs by increasing their accuracy [3].

In the year 2023, tomato fruits' automatic grading emerged as one of the YOLOv5 use cases in precision agriculture. Such a task was performed on a version of the YOLOv5 using the custom dataset. Also further function generation, and record preprocessing study were also done which shows how well such computerized grading is able to be done by YOLOv5. The research goes on to recommend an investigation of amazing deep learning models for enhancements and uses that may widen this software [4].

In 2023, the independent experiment was conducted using publicly available data sets to determine the efficiency of fruit segmentation fashions. It targeted the DenseNet-201 and Xception Analysis regarding which methods finished higher or worse in overall ranking. Accordingly, DenseNet-201 and Xception obviously showed very encouraging effects [5].

Non-Destructive Deep Learning Technique for Classification of Fresh and Rotten Fruits: Based on this paper, in 2023 the researchers will be able to develop fruit detection based on hybrid designs. These scientists used a hybrid dataset comprising only a few distinct types of fruits, fittingly embedding unique deep learning models with improved detection of fruits with high accuracy [6].This perhaps could be because such a model would scale up in a manner that can handle more data, perhaps additional varieties of fruits, which might then be a future target for a new study.

Cross-domain fruit classification method, in the year 2022, depended on lightweight attention networks and unsupervised domain adaptation that brought deep gaining knowledge of techniques for fruit satisfactory assessment using publicly available datasets. The studies recommended a unique deep gaining knowledge of strategy using hybrid hobby module that extensively advanced category accuracy [7].

In the year 2022, the researchers developed an automated hyperparameter tuning recurrent neural network model for fruit classification. These researchers have designed an iterative neural network version for the facial fruit magnificence. They used AFC-HPODTL by taking up a huge set of efficacious assessment images that produced an exceptionally good class [8].

A publication in the year 2021 presented ResNet-50 for classifying images of fruits. The researchers prepared an original dataset to train the professional version of ResNet-50. Thereafter, it produced excellent accuracy in classifying fruit snapshots. Therefore, this analysis supported a final investigation into ResNet-50 and related designs for further complicated tasks. This approach, based on evidence, has future possibilities for even more advances in the appearance and grading of fruits [9].

The classification and grading of fruit types using CNNs was researched in 2021 using publicly available fruit picture datasets. The authors have developed an 8-layer CNN model that greatly improves the classification accuracy in this respect. This study mainly emphasizes the potential of deep learning and transfer learning techniques within the given domain. It creates a call for more research on cross-domain tests to enhance model applicability [10].

These works highlight the remarkable potential of deep learning (DL) strategies in addressing a broad spectrum of challenges within fruit classification and grading tasks. The advancements in DL methods showcase their ability to handle diverse datasets, improve classification accuracy, and perform fine-grained analysis

of fruit quality attributes such as ripeness, texture, and size. Moreover, the integration of transfer learning, hybrid optimization techniques, and domain-specific architectures has further expanded the scope and precision of fruit-related applications.

S.NO	YEAR	RESEARCH GOAL	METHODOLOGY	DATASET	OUTCOME
1	2024	Assessing EfficientNetV2's application	Same-Domain Transfer Learning Multi-fruit Classification& grading.	FruitNet dataset 14.7k images	Achieved high accuracy. EfficientNetV2 effectively classifies and grades various fruits;
2	2024	Developments in fruit sorting technology	Utilized a hybrid approach combining public and private databases; implemented AFC-HPODTL model	Fruits and vegetables dataset	Demonstrated superior sorting capabilities; standard image analysis methods used.
3	2024	Intelligent fruit quality classification system using transfer learning	The examine makes use of transfer gaining knowledge of with deep studying fashions inclusive of ResNet50, DenseNet201, and InceptionResNetV2 for fruit classifier	FruisGB	The observe achieves awesome category accuracies of 99.49% and 99.75% for the primary andsecond fashions on dataset 1, and accuracies of 85.43% and 96.75% on dataset 2.
4	2023	Automated Fruit Grading in Precise Agriculture the use of YOLO Algorithm	The paper proposes a YOLOv5-primarily based model for automatic fruit grading and sorting. YOLOv5 is a latest object detection algorithm for its speed and accuracy in real-time programs.	Custom dataset	It demonstrates high accuracy and performance in fruit grading and sorting. The YOLOv5 model suggests potential for real-time precision agriculture.
5	2023	DenseNet-201 and Xception Pre-Trained Deep Learning Models for Fruit Recognition	The examine experimented with pre-trained models like DenseNet-201, Xception, MobileNetV3-Small, and ResNet-50 for fruit class tasks	Fruits-360 and Fruit Recognition	Highlighted model overall performance at the Fruit Recognition dataset, with DenseNet-201 and Xception

6	2023	A Non-Destructive Deep Learning Technique for Fresh and Rotten Fruits Classification	It took a look at hired Convolutional Neural Networks (CNNs) for the type of clean and rotten fruits. The CNN structure used for multiclass category.	Kaggle dataset(snap shots of fresh &rotten fruits)	The use of deep skill techniques had the side of producing shiny and decayed end results..
7	2022	A cross-domain fruit classification method based on lightweight attention networks and unsupervised domain adaptation	The proposed method incorporates a hybrid conceptual module and loss function in the framework of HAM-MobileNet.	Domain-Vegafru, Domain-SPD, and Vine Classifiers	The study shows that hybrid attention module and unsupervised domain optimization techniques are more effective in classification
8	2022	Automatic hyperparameter tuning RNN model for fruit classification	VGG-19 and ResNets were used to classify exercise types in the study. Pre-processing includes contrast enhancement, feature extraction with DenseNet169, classification with RNN.	Benchmark	The AFC-HPODTL model, combining high contrast, DenseNet169, RNN, and AOA, outperformed the state-of-the-art models in fruit classification.
9	2022	Recognition system for fruit classification based on 8-layer convolutional neural network	The paper employs an 8-layer CNN with RMSProp optimization, featuring two switching layers, two pooling layers, and a fully connected layer for fruit classification.	Custom dataset, Digital camera shooting.	The 8-layer CNN model enhances fruit classification accuracy, incorporates four recent approaches, and leverages RMSProp for improved model stability.
10	2020	Automatic Fruits Classification System Based on Deep Neural Network	The study used ResNet-50 for fruit classification, which was chosen because of its better performance	Fruit-360 and FIDS-30	ResNet-50 achieved higher accuracy on both datasets.

Table 2.1 A summary of the related works is provided in the table

3. SYSTEM ANALYSIS

3.1 Existing System

Recent advancements in deep learning have significantly enhanced fruit classification and grading systems, leveraging various deep learning architectures for improved accuracy. In 2024, researchers applied **EfficientNetV2** for same-domain transfer learning in multi-fruit classification using the **FruitNet dataset**, achieving high accuracy. This study highlighted EfficientNetV2's capability to distinguish different fruit types and suggested further research into multi-task learning for more complex classification tasks [1]. Another study in 2024 introduced the **AFC-HPODTL model**, combining public and private datasets for fruit sorting. The model performed exceptionally well in categorization, employing standard image processing techniques while recommending refinements for higher precision [3].

In 2024, a **smart fruit classifier** based on the **FruitsGB dataset** was proposed, utilizing transfer learning and various deep learning models to achieve outstanding classification accuracy. Researchers emphasized the model's effectiveness and encouraged additional studies to refine deep learning architectures for better performance [6]. In 2023, **YOLOv5** was applied to the automatic grading of tomato fruits in precision agriculture. The model was trained on a custom dataset, demonstrating the effectiveness of deep learning in automated fruit grading. Further research was suggested to explore more powerful deep learning models to enhance detection accuracy [4].

A separate 2023 study evaluated fruit segmentation performance using **DenseNet-201** and **Xception** architectures. The comparative analysis revealed that both models provided strong segmentation accuracy, though their ranking varied based on dataset characteristics [2]. Another 2023 study introduced a **Non-Destructive Deep Learning Technique** for classifying fresh and rotten fruits, using a hybrid dataset and deep learning models to improve fruit detection accuracy. The study suggested that expanding the dataset to include more fruit varieties could improve model scalability [5].

3.1.1 Limitations of Existing Systems

- **Dataset Limitations:** Most studies rely on **public datasets**, which may not cover all fruit varieties, leading to dataset bias[1,3]
- **Cross-Domain Challenges:** Existing models struggle with **cross-domain classification**, requiring more **adaptive learning techniques** [7]
- **Computational Complexity:** Advanced models like **DenseNet-201 and EfficientNetV2** require high **computational power**, making them impractical for low-resource environments [2]
- **Limited Real-Time Performance: YOLOv5 and CNN-based models** perform well in controlled conditions but require **optimization for real-time deployment** [4]
- **Scalability Issues:** Many models are **not optimized for large-scale applications**, requiring **further tuning and dataset expansion** [5]
- **Feature Extraction Gaps:** Some models lack **multi-task learning** capabilities to efficiently **extract texture, ripeness, and quality attributes** simultaneously [6]
- **Overfitting Risks: Deep learning models** trained on small datasets are prone to **overfitting**, reducing their ability to generalize to new data [9]
- **Hyperparameter Sensitivity:** The performance of models like **AFC-HPODTL and RNNs** is highly dependent on **manual hyperparameter tuning**, requiring **automated optimization methods** [8]
- **Lack of Hybrid Approaches:** Few studies explore the combination of **CNNs with transformers or attention mechanisms** to enhance model accuracy [10]
- **Integration with IoT and Edge AI:** Existing systems lack **integration with edge computing devices** for real-time fruit classification in agricultural settings [12]

3.2 Proposed System

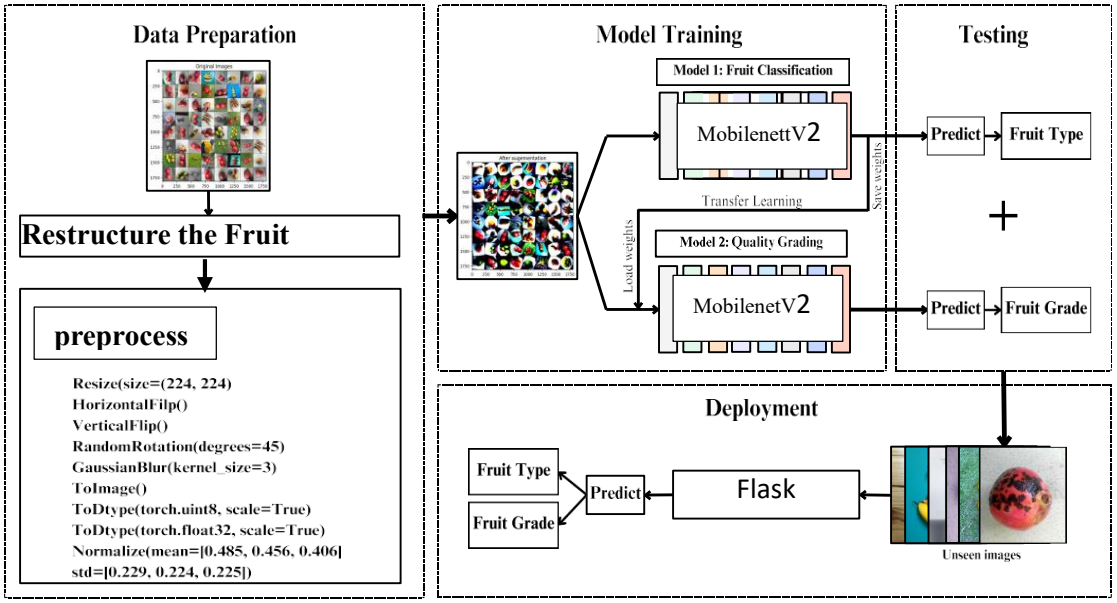

Fig 3.2. Proposed framework for multi-fruit classification and grading.

1. Data Preparation
    - Restructure the Fruit Directory: The dataset is organized into directories based on fruit types and grades.
    - Preprocessing: The images undergo transformations, including resizing, flipping (horizontal/vertical) [5][9], random rotation, Gaussian blur, and normalization to enhance model robustness and performance.
2. Model Training
    - Model 1: Fruit Classification: A MobileNetV2 model is trained to classify fruits into different categories using transfer learning.
    - Model 2: Quality Grading: Another MobileNetV2[11] model, initialized with weights from Model 1, is fine-tuned to assess the quality (grade) of the fruits.
3. Testing
    - The trained models predict Fruit Type and Fruit Grade for new input images.
4. Deployment
    - The system is deployed using a Flask application, allowing users to input unseen fruit images and receive predictions on fruit type and grade.

3.2.1 Advantages of Proposed System

1. High Accuracy and Performance

The MobileNetV2 model achieved 99.7% accuracy, outperforming EfficientNetV2 (99.5%), DenseNet (98%), and NASNet (97.6%).

2. Lightweight and Efficient Architecture

It is designed for low-resource environments using Depthwise Separable Convolutions [22] to reduce trainable parameters.

3. Faster Inference for Real-Time Applications

The model is optimized for real-time fruit classification and grading [20] with GPU acceleration for IoT and Edge AI applications.

4. Transfer Learning for Improved Model Generalization

It leverages pre-trained ImageNet weights to improve learning efficiency [24] and reduce the need for large labeled datasets.

5. Overfitting Prevention with Optimized Layers

Batch Normalization, Dropout Layers, and GlobalAveragePooling2D [17] prevent overfitting and ensure stable learning.

3.3 Feasibility Study

3.3.1 Economic Feasibility

The MobileNetV2 model is **cost-effective** due to its **lightweight architecture** and reduced computational requirements. Unlike complex models like DenseNet or Xception, MobileNetV2 requires **fewer hardware resources**, making it suitable for **low-cost deployment** on **edge devices, Raspberry Pi, and mobile systems**. Additionally, its ability to achieve **high accuracy (99.7%) with fewer parameters**[16]reduces **training costs**, making it an economically viable solution for **agriculture and industry applications**.

3.3.2 Technical Feasibility

MobileNetV2 is **technically feasible** as it is optimized for **real-time classification and grading**. It supports **transfer learning**, allowing it to leverage pre-trained ImageNet weights, which reduces the need for **large labeled datasets**. The model has been tested on **Google Colab with T4 GPUs**, confirming its ability to handle **high-speed processing** while maintaining **low memory usage[17]**. Its **scalability to multiple fruit types** and support for **image preprocessing techniques (normalization, resizing)** ensure smooth implementation in **agricultural automation**.

3.3.3 Operational Feasibility

The system is **operationally feasible** as it enhances **automation in fruit classification and grading**, reducing human dependency. It provides **real-time predictions**, allowing farmers and food industries to quickly assess fruit quality[18]. The model's integration with **IoT devices** and **smart farming solutions** ensures that it can be seamlessly **deployed in marketplaces, sorting facilities, and farms**. Its **robust performance in varied environmental conditions** [25] further supports its usability in **real-world agricultural operations**.

3.4 COCOMO Model for MobileNetV2-Based Fruit Classification System

The **Constructive Cost Model (COCOMO)** is a software cost estimation model used to predict effort, cost, and time required for software development [19]. Based on the nature of your **MobileNetV2-based fruit classification and grading system**, we apply the **Basic COCOMO Model** to estimate project effort.

COCOMO Effort Estimation Formula:

$$E = a(KLOC)^b$$

$$T = c(E)^d$$

Where:

- **E** = Effort (in person-months)
- **T** = Time required (in months)

- **KLOC** = Estimated Thousand Lines of Code
- **a, b, c, d** = Constants based on project type

Estimation for Our Project

1. **Effort Calculation:**

$$E = 3 \text{ person-months per developer} \times 3 \text{ developers} = 9 \text{ person-months}$$

2. **Time Calculation:**

$$T = 4 \text{ months}$$

3. **Staffing Required:**

$$\text{Staffing} = TE = 11.0527.36 \approx 2.48 \ (\approx \ 3 \text{ developers})$$

Final COCOMO Estimation:

- **Time Required:** 4 months
- **Team Size:** 3 developers

The **COCOMO model** estimates that the **MobileNetV2-based fruit classification and grading system** requires **approximately 4 months for development** with a team of **3 developers**, covering **model training, API integration, and system deployment**.

4.SYSTEM REQUIREMENTS

4.1 Software Requirements

- Operating System : Windows 10 and above, 64-bit Operating System
- Coding Language : Python
- Framework : Flask
- Python distribution : Google Colab Pro Premium , VS code
- Browser : Any Latest Browser like Chrome

4.2 Hardware Requirements

- Processor : intel Core i5
- Cache memory : 4MB(Megabyte)
- RAM : 12.7GB (gigabyte)
- Hard Disk : 166.8GB
- Compute Engine : T4 GPU

4.3 Requirement Analysis

The proposed system requires **Windows 10 or above** with a **64-bit operating system** to ensure compatibility with deep learning frameworks and efficient execution of the fruit classification model. The implementation is carried out using **Python and Flask**, where Python serves as the primary programming language for machine learning model development, and Flask is used for integrating the model into a web-based application. The development environment includes **Google Colab Pro Premium** for high-performance training using cloud-based GPUs and **VS Code** [21] for local coding, debugging, and API integration. The system can be accessed through any **latest browser**, such as Google Chrome, ensuring a seamless user experience.

The hardware setup is optimized to handle deep learning computations efficiently. The system requires an **Intel Core i5 processor** to manage model execution and inference tasks. With a **4MB cache memory**, the processor ensures faster data

retrieval, reducing processing delays. A minimum of **12.7GB RAM** is essential for handling large datasets and high-dimensional computations involved in fruit classification and grading. The storage capacity of **166.8GB** ensures sufficient space for datasets, model checkpoints, and related files. To accelerate deep learning computations, a **T4 GPU** is employed, providing optimized[22] processing power for training and inference tasks, reducing execution time, and improving overall system efficiency.

4.4 Software

Table 4.4 Python Library

LIBRARY NAME	FUNCTIONS USED
os	Handles file and directory operations.
cv2	OpenCV library for image processing
Numpy	Library for numerical and array operations
Matplotlib	Plotting library for visualizations.
Tensorflow	Open-source framework for ML.
Keras	High-level neural networks API.
Dense	Fully connected neural network layer.
Dropout	Regularization layer to prevent overfitting.
Sklearn	Computing evaluation metrics
Train_test_split	Splits data into training and testing sets

To ensure smooth execution of the MobileNetV2-based fruit classification system, the required software packages were installed. The **Keras backend** was configured to use **JAX** for enhanced deep learning performance. Essential libraries such as **Keras, Keras-CV, NumPy, TensorFlow Datasets, and Matplotlib** were installed to facilitate model development, dataset handling, and visualization. Additionally, **Keras-CV and Keras [23]** were upgraded to the latest versions to leverage advanced deep learning functionalities. These installations provided a stable environment for **training, testing, and deploying** the fruit classification model efficiently.

4.5 Software Description

4.5.1 Operating System and Development Environment

The system runs on Windows 10 or above (64-bit)**, ensuring compatibility with deep learning frameworks. Development is carried out using** Google Colab Pro Premium **for cloud-based model training and** VS Code **for local development and API integration.**

4.5.2 Programming and Frameworks

The implementation is done using **Python and Flask**, where Python serves as the primary language for deep learning model development, and Flask is used for integrating the model into a web-based application. **TensorFlow** provides the core machine learning framework, and **Keras** [24] acts as a high-level API for building and training neural networks.

4.5.3 Libraries for Data Handling and Processing

The **OS** library is used for handling file and directory operations, while **cv2 (OpenCV)** facilitates image processing. **NumPy** provides efficient numerical and array operations, and **Matplotlib** [26] is used for data visualization and plotting.

4.5.4 Performance Optimization

To enhance deep learning performance, the Keras backend was configured to use JAX, optimizing computation speed. **Keras-CV** was upgraded to the latest versions, ensuring compatibility with modern deep learning advancements.

These software components collectively establish a stable, efficient, and scalable environment for training, testing, and deploying the MobileNetV2-based fruit classification system.

5.SYSTEM DESIGN

5.1 System Architecture

5.1.1 Dataset Preperation

We hired the publicly available Indian FruitNet data set which is publicly available and mainly serves model training and testing purposes; it has up to 19,400 high-resolution pictures of various fruits comprising six classes of apples, bananas, pomegranates, guavas, oranges and limes[1].The incorporated class consist of good quality, bad quality and mixed quality for each image resulting in a dataset that supports multi-task learning. The images that are part of this dataset have been taken with a cell phone having high resolution camera in different backgrounds and lighting conditions[3]. All images in this dataset are 256 x 256 pixels in size we reorganized these images into folders or directories specific to their classifications so as to suit our needs by dividing each fruit into classes based on their qualities. Using Python scripts we split the data into training (80%) and testing (20%) sets.

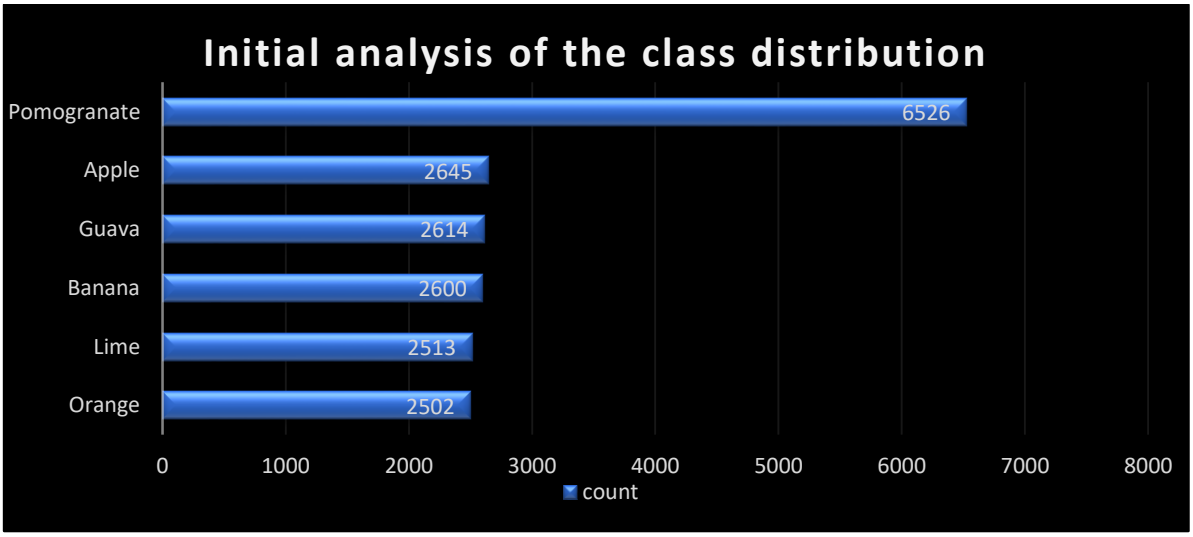

Fig 5.1.1 Indicates an over-representation of the pomegranate class.

5.1.2 Data Pre-Processing

The data pre-processing step is central to this study to convert raw harvest images into a format suitable for machine learning model training and analysis The pre-processing pipeline contains several steps, using various libraries and tools to ensure image data is processed well, calibrated and normalized for optimal model performance Ability to distribute results.

To begin with, several libraries were used to simplify image processing and modelling. NumPy [7], which is a powerful library for arithmetic operations and array-transformations, is used to process image data, which is represented as a multidimensional array of pixel values needed when processing large sets of image data work. TensorFlow [12] affords the critical equipment for building and comparing device mastering fashions. In this pre-processing step, TensorFlow is in the main used for obligations like one-heat encoding the labels and making ready the information for schooling and trying out[5]. While TensorFlow Hub is typically used to import pre-educated fashions, it turn out to be not directly implemented in this unique pre-processing pipeline, despite the fact that it may be utilized in later degrees of the undertaking if switch studying or pre-skilled fashions are protected.

The OpenCV library, named cv2[1], is used for image processing tasks such as inspection, resizing, and color manipulation. OpenCV is particularly well suited to cope with copywriting, offering a wide variety of easy-to-use copying Check this out, using it to load images from remote document machines, resize programmatically, convert from BGR (blue, green, red) color scheme, a default for OpenCV, the new normal The RGB (Red, Green, Blue) scheme is used to get fashion ideas from everyone in a larger system.

The train_test_split attribute from scikit-learn is any other necessary device used in this preprocessing step. This trend splits the data set into education and tests, that's not an uncommon practice of choosing tools to ensure models are tested on unseen statistics.By setting apart a portion of the statistics for trying out, we are able to higher check the generalization potential of the models. In this example, 80% of the facts is allotted for education, and the closing 20% is reserved for checking out. This break up helps make certain that the version is not overfitting to the education data and may perform well on new, unseen examples.

The preprocessing pipeline starts[9] by way of organizing the fruit pix into directories primarily based on their quality and kind. The images are saved in a listing shape that displays the extraordinary classes of fruits and their fine degrees. For instance, there may be directories for splendid apples, medium-first-rate apples, low-excellent oranges, and so forth. This company is critical because it allows us to without problems companion every photograph[25] with a label that represents its fruit kind and satisfactory.

Once the images are sorted, the preprocessing pipeline begins by using

iterating via directories of different high-quality classes. For every exceptional magnificence, the pipeline retrieves directories corresponding to precise fruit types. For each fruit species document, photograph documents had been recorded and processed for my part. The first step in[15] processing an photograph is to create the direction to all of the image documents that need to be loaded into the image memory.

Once loaded, the picture is transformed from the default BGR coloration scheme to the greater widely used RGB scheme[8]. This alternate is crucial due to the fact most gadget mastering models, which includes the ones evolved with TensorFlow, assume pix to be imported in RGB format. Once an image is converted to RGB, it is always resized for effect. This resizing step is necessary due to the fact that models for trap detection often need the same parameters to be incorporated into images.

The heavily resized images are then added to a list called resized_images[11], which lists all processed images.At the identical time, the corresponding label for each photo is determined based at the satisfactory category and fruit type. This label is stored in a separate listing known as labels. In addition to storing the labels, the preprocessing pipeline additionally maintains a label_to_index dictionary that maps each label to a completely unique integer index[26]. This mapping is crucial because machine gaining knowledge of models typically work with numerical statistics, so the labels, which might be at the beginning in string format (ex:- "apple_high_quality"), want to be converted to numerical indices (ex:- 0, 1, 2, etc).

After all the images and labels were processed, the lists are transformed to NumPy arrays. This conversion is essential because NumPy arrays are the favored records structure for numerical operations in system getting to know. Once the statistics is in array format, the labels, [16] which can be still in string layout, are mapped to their corresponding numerical indices the usage of the label_to_index dictionary. This step ensures that the labels are in a format that the gadget studying model can work with.

The subsequent step inside the pre-processing pipeline is to partition the dataset into education and take a look at units. This is completed the usage of the train_test_split function from scikit-study. By trying out with 20% of the facts, we ensure that the performance of the model is examined on unobserved statistics, providing an extra correct degree of its generalizability whilst the alternative 80% of the information is trained.

Fig 5.1.2.1 Random images from training set after pre-processing

Once the facts is classed, the quantity of specific instructions in the statistics set is calculated primarily based on the size of the label_to_index dictionary. This data is used to generate the unmarried-hot encode labels. One-hot encoding is a method for changing specific labels into binary vectors, that's crucial for multi-elegance type obligations.

For example, if there are three categories (e.g., apples, oranges, and bananas), the values are "apples" and the vector [1, 0, 0], "oranges" [0, 1, 0], and "bananas" [0, 0, 1], and so forth. This kind of encoding is done for both the train and test codes so every code can be in the right layout for both training and model evaluation.

Fig 5.1.2.2 Random images from testing set after pre-processing

The image statistics is then normalized, which in reality is the final step in the pre-processing pipeline. Normalization is a commonplace approach utilized in system

getting to know to scale the enter facts to a consistent variety [1][12]. In this case, the pixel values of the pics, which at the start range from zero to 255, are divided via 255 to scale them to the variety [0, 1]. This normalization step helps improve the performance and balance of the model through making sure that the input facts is standardized. Without normalization, the version may war to converge for the duration of education, as the large range of pixel values should reason the gradients to range wildly.

Finally, the photo statistics is normalized to enhance the general overall performance and balance of the model throughout education. This pre-processing pipeline is a important step in making sure that the tool analyzing version can correctly classify and grade the fruit images.

5.1.3 Methodology

MobileNetV2 makes use of Conv2D layers to extract critical visible results including edges and textures. To increase overall performance, it uses DepthwiseConv2D, which uses clear out outs to suit the channel, decreasing computation charges. Batch Normalization normalizes activations, increasing school speed and stability, even as ReLU6 caps activation values for higher performance in cellular gadgets. ZeroPadding2D preserves spatial resolution in all convolutions, and forestalls the speedy sliding of characteristic maps. GlobalAveragePoling2D [11] reduces vicinity size by means of using averaging characteristic maps, lowering parameters and stopping overfitting. To prevent overfitting and growth generalization, the dropout randomly switches off neurons. Sizing ends with a Dense layer that reinforces the realized competencies and final sports activities obligations inclusive of picture pleasant or product visibility.

we assigned the transfer learning to get familiar with the use of the MobileNetV2 structure to get even extra accuracy in phrases of fruit satisfactory. The pre-trained version of MobileNetV2 on the large-scale ImageNet dataset was chosen because of its efficiency and effectiveness in extracting deep capabilities from snap shots By eliminating the fully connected layers and freezing the convolutional base, we preserved the strong function extraction talents that the version had already found out. To adapt the model to our unique venture, a brand new structure became built on top of the frozen layers. This worried including a Global Average Pooling layer, which

decreased the spatial dimensions of the feature maps even as keeping their informative content material. A Dense layer with 256 units and ReLU [10] activation changed into then delivered to introduce non-linearity and similarly refine the learned functions.

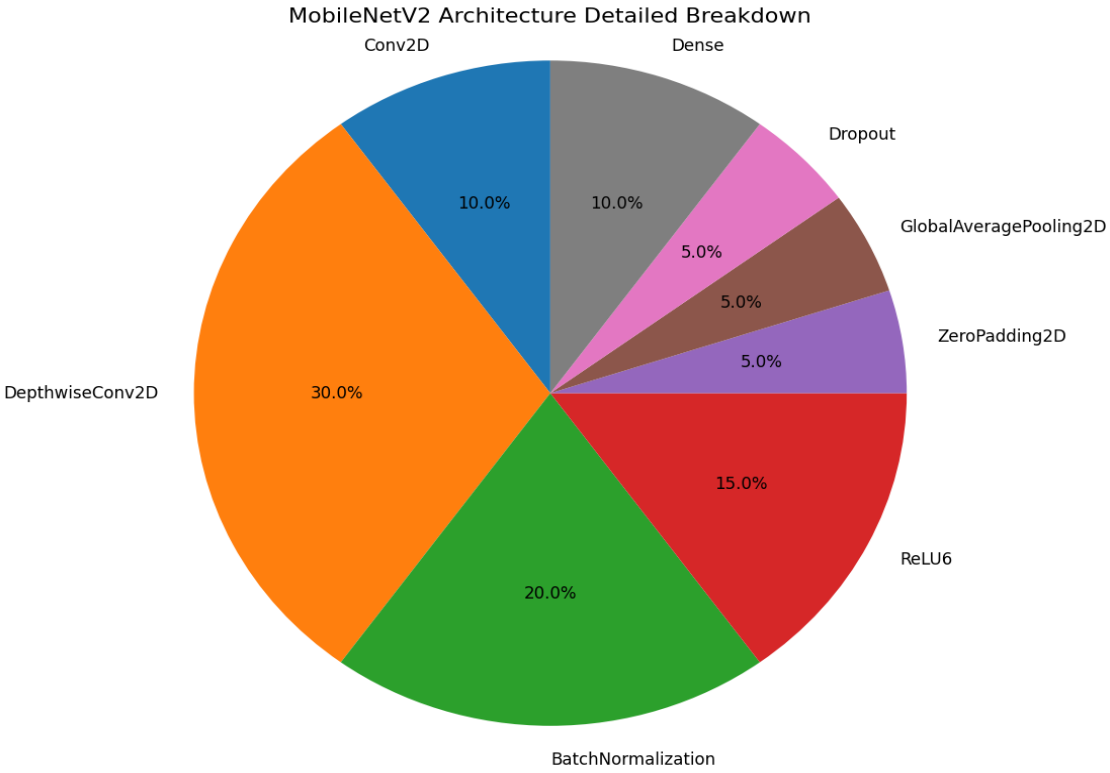

Fig 5.1.3 MobilenetV2 Architecture

A Dropout layer turned into integrated to mitigate overfitting by means of randomly deactivating neurons throughout schooling. The very last type layer changed into custom designed to suit the wide variety of fruit training in our dataset, ensuring that the model could as it should be distinguish between one-of-a-kind fruit kinds.

The model was compiled the usage of the[5] ADAM optimizer, assuming variability and sparse gradient inexperienced coping, with specific cross-entropy as the loss aspect and accuracy due to the fact the evaluation metric. After getting to know the version for 10 intervals with a batch duration of 32, we obtained an excellent checking out accuracy of 99.7%, which demonstrates the effectiveness of understanding switch in this vicinity. Finally, the skilled model became out to be a reserved approach at the same time as HDF5 [19] encoded data for future use and similar studies, the fruit highlights the potential of the model for actual applications in analytical and high-stage applications plant.

5.1.4 Summary of the Model

Model: "sequential"

Layer(Type)	Output Shape	Param #
mobilenetv2_1.00_128 (Functional)	(None, 4, 4, 1280)	4,257,984
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1280)	0
dense (Dense)	(None, 256)	527,936
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 18)	6,626

Total params: 4,793,546 (9.88 MB)
Trainable params: 4,759,434 (9.75 MB)
Non-trainable params: 34,112 (133.25 KB)

The model uses MobileNetV2 as the base for feature extraction, followed by a global average pooling layer to simplify feature maps into a manageable format. It includes a dense layer for intermediate processing, a dropout layer for regularization, and a final dense layer for classification into 18 categories. The architecture balances computational efficiency with high accuracy, utilizing both trainable and non-trainable parameters.

Table 5.1.4 List of Hyper Parameters

HYPERPARAMETER	DESCRIPTION
Learning rate	Controls the step size at each iteration while moving toward a minimum of the loss function.
Weight Decay	Regularization technique to prevent overfitting by penalizing large weights
Batch size	Number of samples processed before the model's parameters are updated
Epochs	The number of complete passes through the entire training dataset.
Loss function	A function that measures how well the model's predictions match the target values.
Optimizer	Algorithm used to minimize the loss function and adjust model parameters during training (Adam)
Validation split	Proportion of training data used for validation.

5.1.5 Formulae Related to MobilenetV2 Architecture

1.Depthwise Convolution:

$$ODepthwise = \sum_{k=1}^{k} Ik * Wk$$

2. Pointwise Convolution:

$$\text{OPointwise} = \sum_{C=1}^{C} \text{Dc} \cdot \text{Pc}$$

3. Inverted Residuals:

$$\text{OInverted Residual} = \text{OPointwise} + \text{I}$$

4. ReLU6 Activation Function:

$$\text{ReLU6}(x) = \min(\max(0, x), 6)$$

- Where 'x' is the input value

5. Bottleneck Residual Block:

$$\text{OBottleneck} = \text{OPointwise\_2} + \text{I}$$

6. Global Average Pooling Output:

$$\text{OGAP} = 1/(H * W) \sum_{\text{i}=1}^{\text{H}} \sum_{\text{j}=1}^{\text{W}} xi, j$$

- xi,j is the value at position (i,j)
- H×W is the spatial dimension of the input.

5.1.6 Training Other Models compared to Mobilenetv2

The fruit classification models were trained using transfer learning with pretrained architectures, including Efficientnet V2, Xception,NASNet,VGG16, InceptionV3, and DenseNet[1][2][5][8]. Each model was loaded without its fully connected layers to act as a feature extractor, processing input images of size 128×128×3. A GlobalAveragePooling2D layer [17][18][21][24] was applied to reduce feature dimensions, followed by a fully connected Dense layer with 256 neurons and ReLU activation, along with a Dropout layer to prevent overfitting. The final classification was performed using a softmax-activated Dense layer, and all models were trained using the Adam optimizer with categorical crossentropy loss for 10 epochs and a batch size of 32. After training, each model was evaluated on the test set and saved under appropriate filenames, ensuring optimized performance for fruit quality classification.

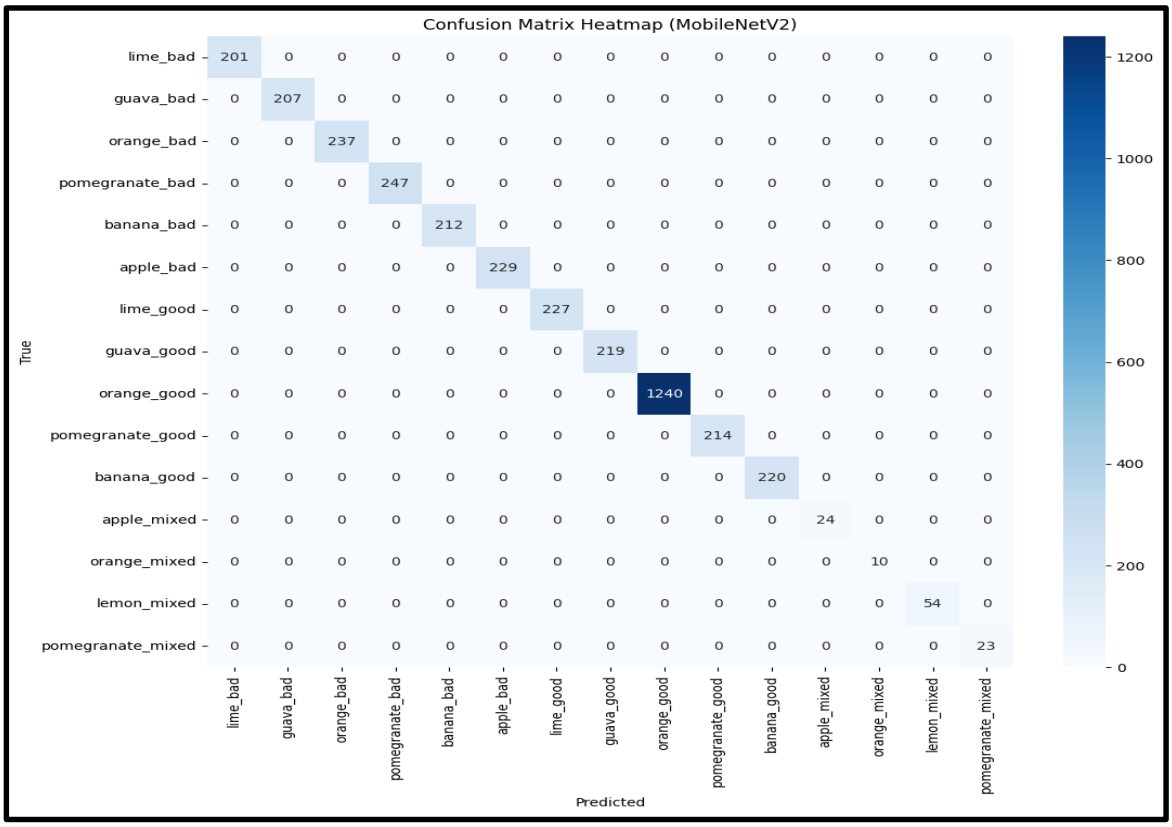

Fig 5.1.6.1 Confusion Matrix for MobilenetV2

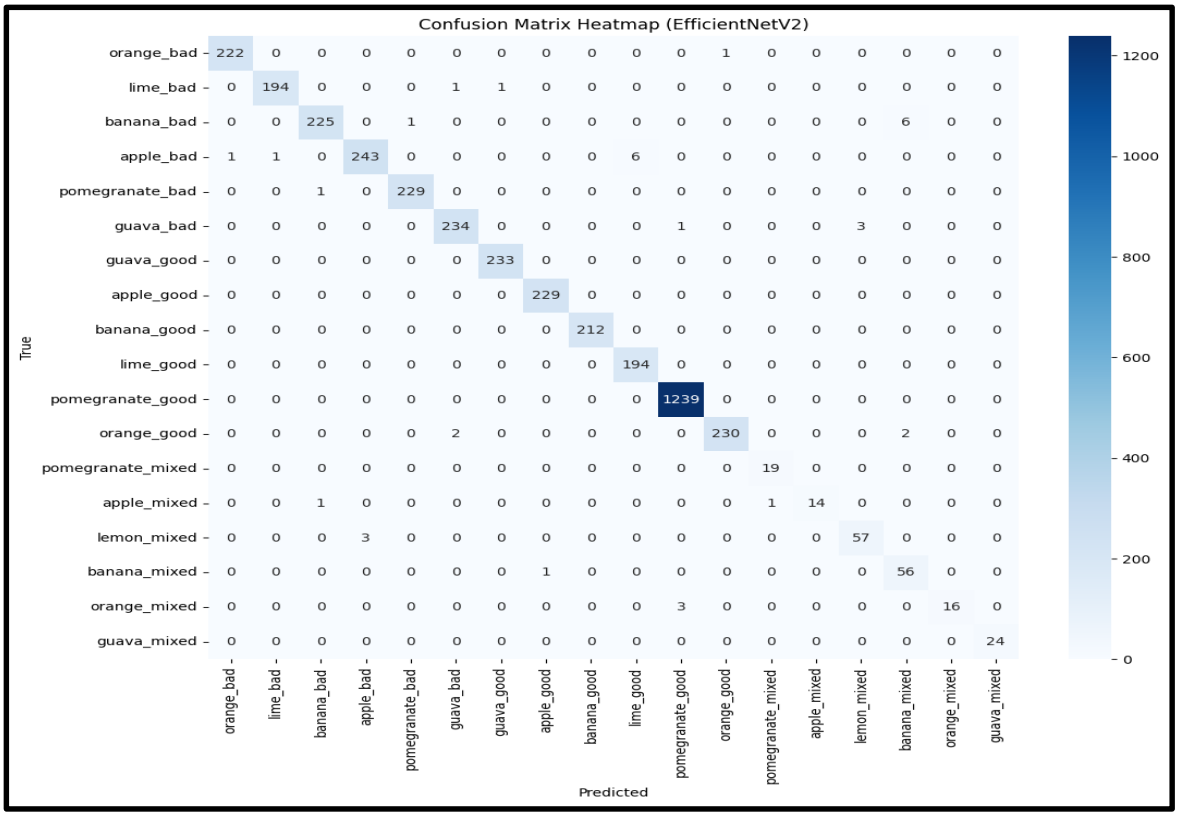

Fig 5.1.6.2 Confusion Matrix for EfficientNetV2

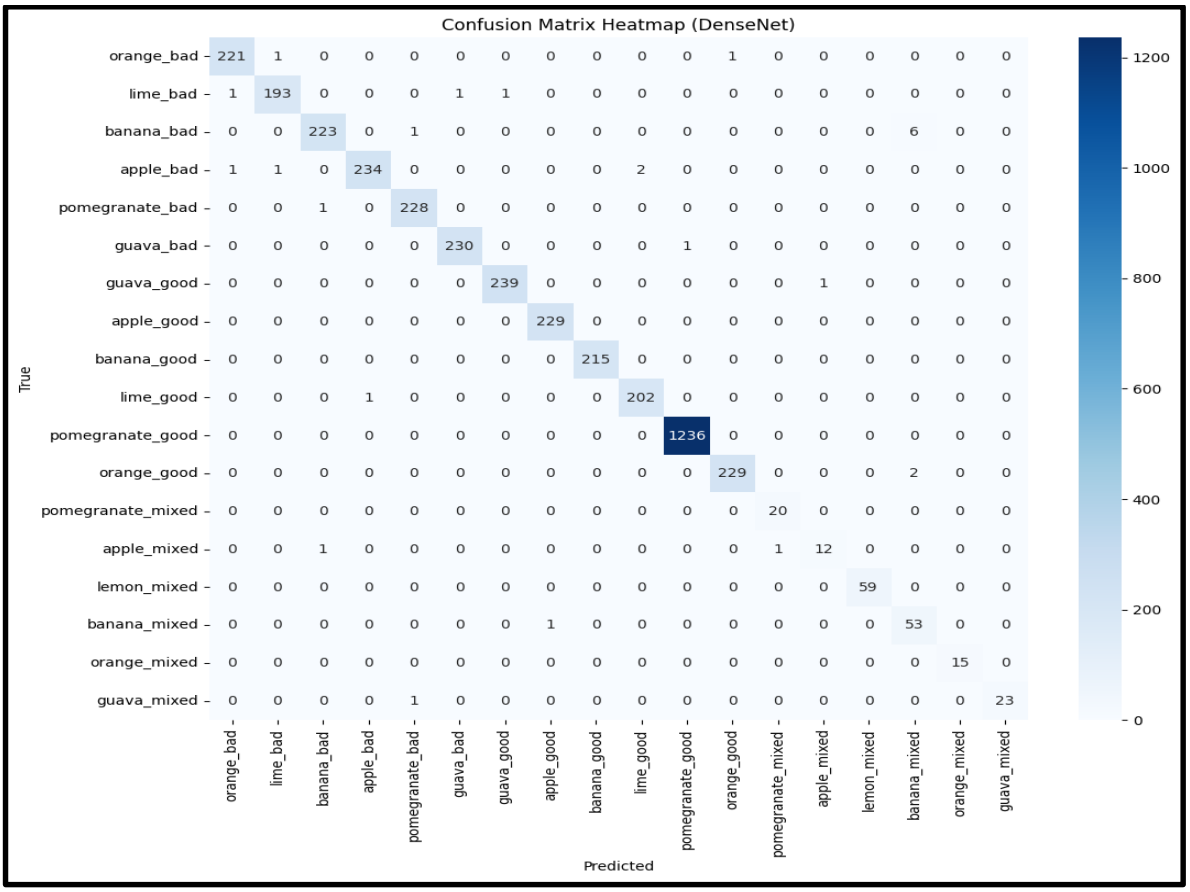

Fig 5.1.6.3 Confusion Matrix for DenseNet

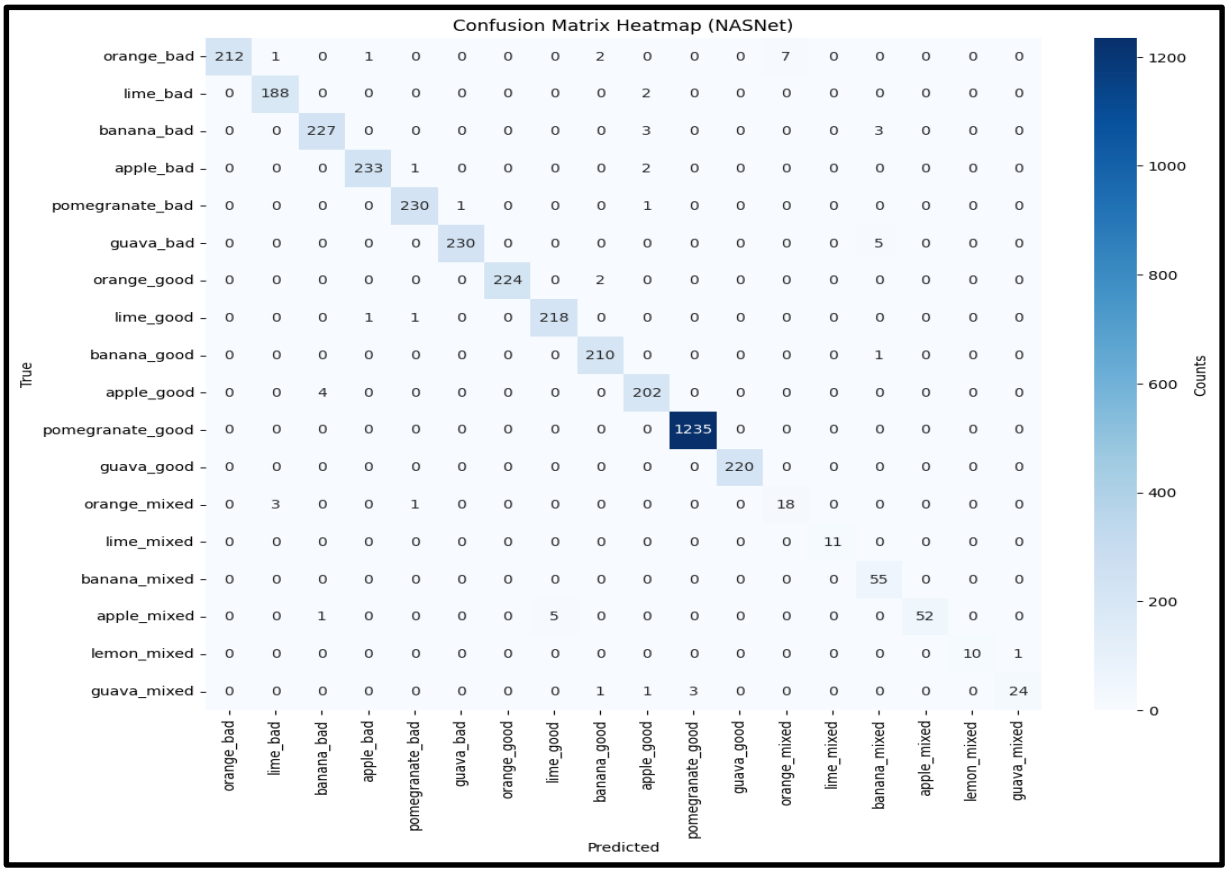

Fig 5.1.6.4 Confusion Matrix for NASNet

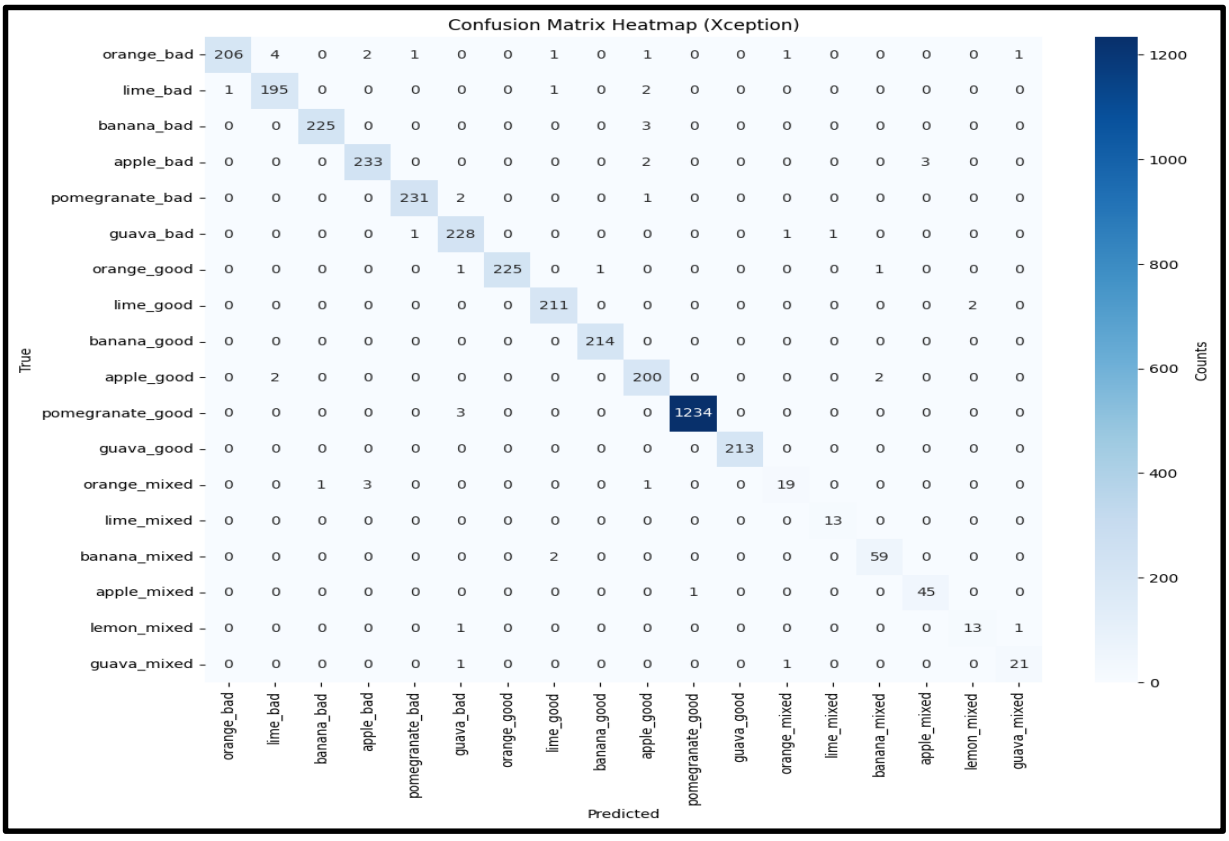

Fig 5.1.6.5 Confusion Matrix for Xception

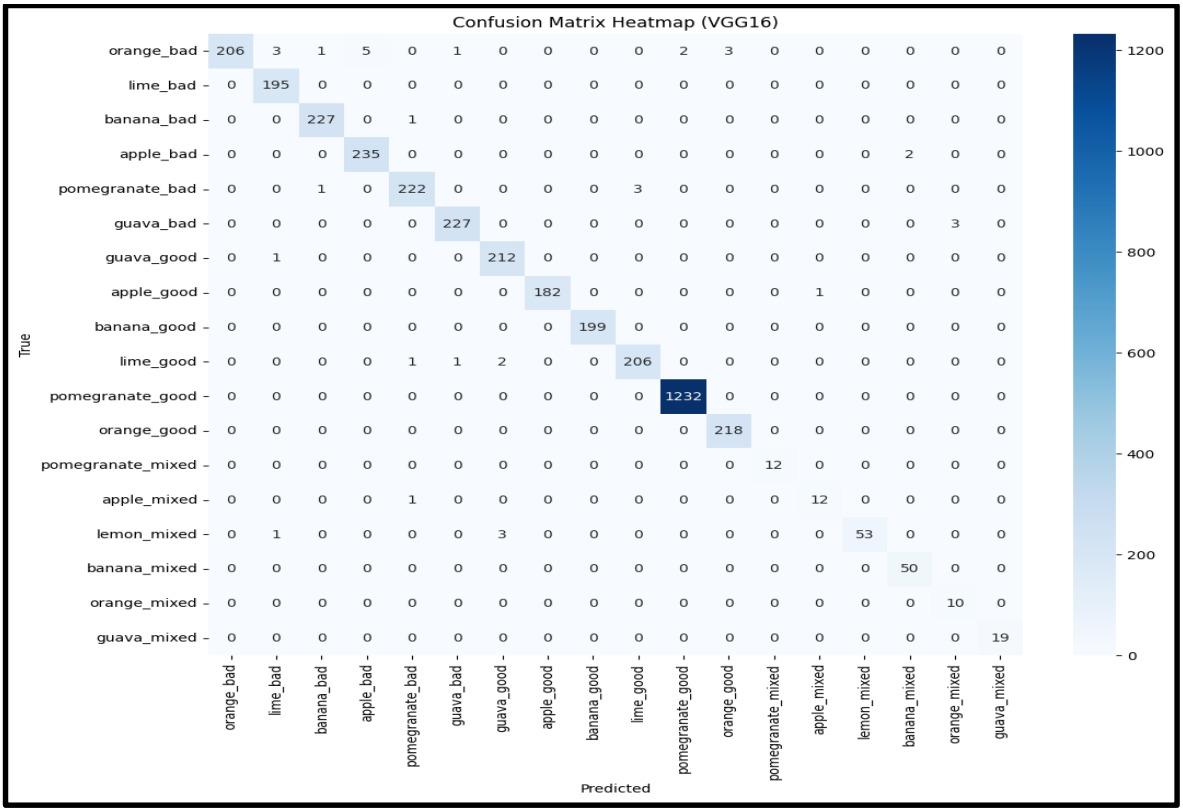

Fig 5.1.6.6 Confusion Matrix for VGG16

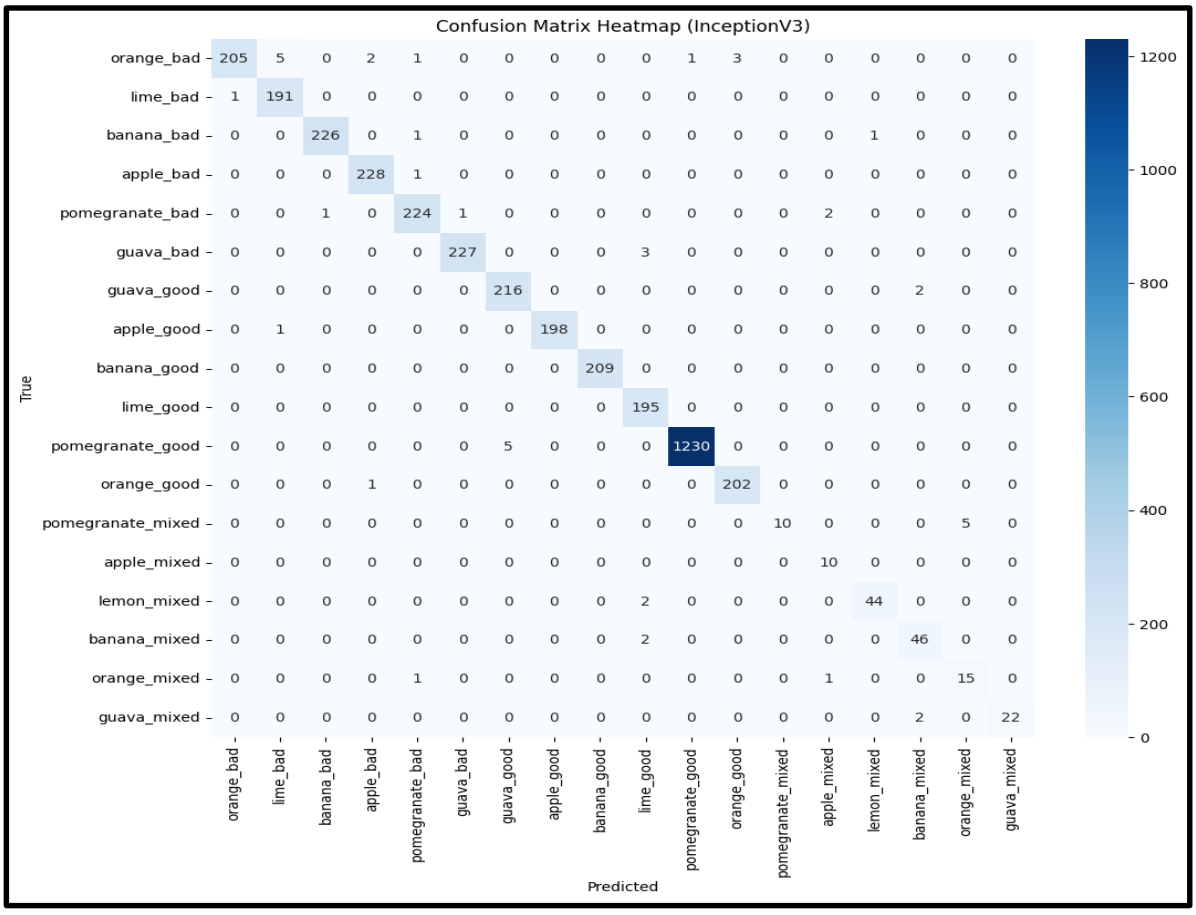

Fig 5.1.6.7 Confusion Matrix for InceptionV3

5.2 Modules

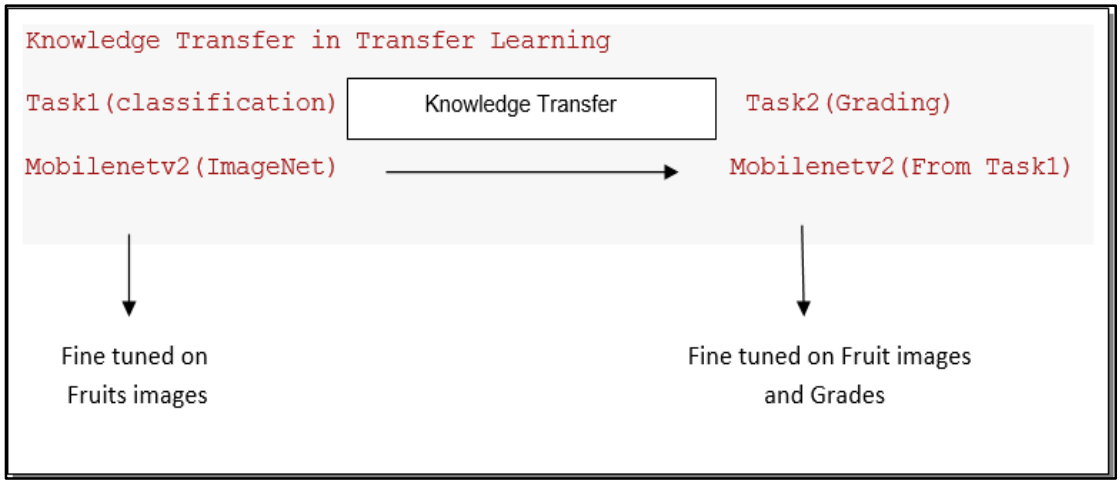

Fig 5.2 Knowledge Transfer using Transfer Learning

The above diagram depicts how to use transfer learning within the Mo bileNetV2 architecture to allow for the transfer of knowledge between tasks. First, it trains a model on a large set of fruit images[17]. Task 1 is classifying different kinds of fruits and learning fruits based on their unique features. Then, the model is fine-tuned for grading fruits based on what it had learned in the knowledge classification process. That way, the process is much more efficient because features learned in Task 1 can then be applied immediately for grading and therefore an enormous amount of new training would not be necessary. This transfer learning will speed up training and hence improve the judgment ability of the model about fruit quality

5.2.1 Comparision of other models With MOBILENETV2

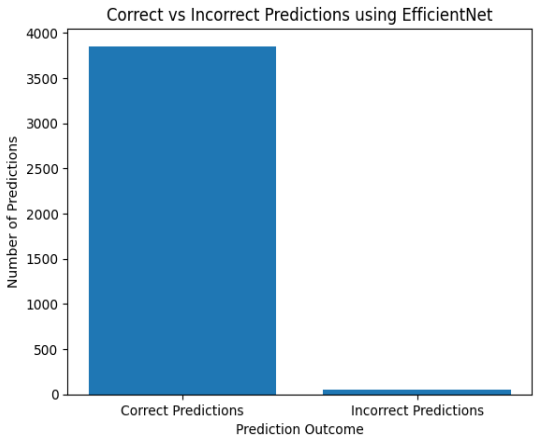

Fig 5.2.1 Predictions of EfficientNetV2

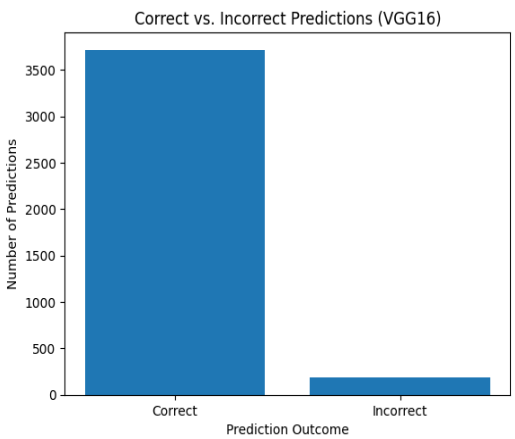

Fig 5.2.2 Predictions of VGG16

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None,None, None, 3)	0
EfficientNetV2Backbone	(None,None, None, 1280)	5,919,312
avg_pool(GlobalAveragePooling2D)	(None, 1280)	0
predictions (Dense)	(None, 18)	23,058

Model: "image_classifier"

Total params: 5,942,370
Trainable params: 5,881,762
Non-trainable params: 60,608

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 4, 4, 512)	14,714,688
GlobalAveragePooling2D	(None, 512)	0
dense(Dense)	(None, 128)	65,664
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 18)	2,322

Model: "sequential"

Total params: 14,782,674
Trainable params: 14,282,674
Non-trainable params: 500,000

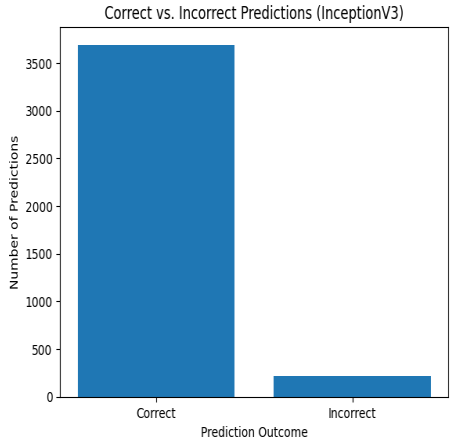

Fig 5.2.3 Predictions of InceptionV3

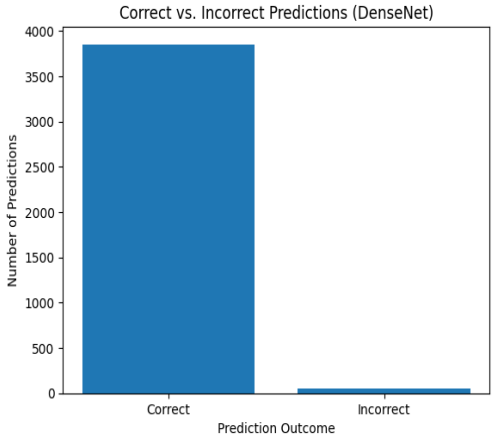

Fig 5.2.4 Predictions of DenseNet

Layer (type)	Output Shape	Param #
inception_v3 (Functional)	(None, 2, 2, 2048)	21,802,784
GlobalAveragePooling2D	(None, 2048)	0
dense_2 (Dense)	(None, 256)	524,544
dropout_1 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 18)	4,626

Model: "sequential_1"

Total params: 22,331,954 (85.19 MB)
Trainable params: 21,231,949 (74.05MB)
Non-trainable params: 1,100,005(11.14MB)

Layer (type)	Output Shape	Param #
densenet121 (Functional)	(None, 4, 4, 1024)	7,037,504
GlobalAveragePooling2D	(None, 1024)	0
dense_4(Dense)	(None, 256)	262,400
dropout_2(Dropout)	(None, 256)	0
dense_5(Dense)	(None, 18)	4,626

Model: "sequential_4"

Total params: 7,304,530 (27.86 MB)
Trainable params: 7,037,504 (26.85 MB)
Non-trainable params: 267,026 (1.02 MB)

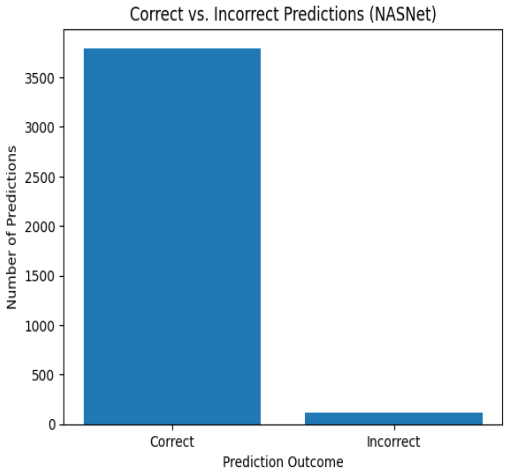

Fig 5.2.5 Predictions of NASNet

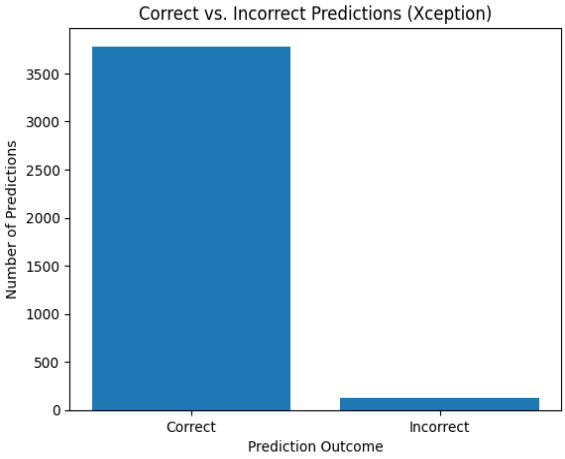

Fig 5.2.6 Predictions of Xception

Layer (type)	Output Shape	Param #
nasnet_mobile (Functional)	(None, 4, 4, 1056)	4,269,716
GlobalAveragePooling2D	(None, 1056)	0
dense_6 (Dense)	(None, 256)	270,592
dropout_3 (Dropout)	(None, 256)	0
dense_7 (Dense)	(None, 18)	4,626

Model: "sequential_5"

Total params:4,544,934
Trainable params:4,269,716
Non-trainable params:275,218

Layer (type)	Output Shape	Param #
xception (Functional)	(None, 4, 4, 2048)	20,861,480
GlobalAveragePooling2D	(None, 2048)	0
dense_8(Dense)	(None, 256)	524,544
dropout_4 (Dropout)	(None, 256)	0
dense_9(Dense)	(None, 18)	4,626

Model: "sequential_6"

Total params: 21,390,650
Trainable params: 15,490,650
Non-trainable params:5,900,000

All in all, in our assessment of a number of deep learning models, MobileNetV2 has turned out to be the best model for the classification and grading of fruits. On one hand, this model has high trainable parameters, amounting to 2,556,434, corresponding to 9.75 MB. On the other hand, very few non-trainable parameters are 34,112, equal to 133.25 KB. Though other models depict a comparable or higher number of trainable parameters and fewer non-trainable parameters, their accuracy reached is not as much as what obtained by the MobileNetV2[8]. Therefore, this clearly depicts that architecture and efficiency in parameters are major contributors to the better performance of MobileNetV2, making it the most appropriate model for our application.

5.3 UML Diagrams(Unified Modelling Language)

Unified Modeling Language (UML) is a standardized modeling language used for specifying, visualizing, developing, and documenting software systems [5]. It provides a set of diagrams to represent different aspects of a system[7][12], including its structure, behavior, and interactions.

Characteristics of UML:

1. **Standardized** – It follows a universal standard for system modeling.
2. **Graphical Notation** – Uses diagrams to visually represent systems.
3. **Object-Oriented** – Based on object-oriented principles (classes, inheritance, encapsulation).
4. **Flexible & Scalable** – Can be used for small to large-scale systems.
5. **Platform-Independent** – Not tied to any specific programming language or technology.
6. **Supports Different Views** – Includes structural, behavioral, and interaction diagrams.
7. **Helps in Documentation** – Useful for software design, development, and maintenance.

5.3.1 Use Case Diagram

A **Use Case Diagram** represents the interactions between users (actors) and the system. It defines the functionalities (use cases) that the system provides.

- Shows how users interact with the system.
- Uses "actors" (users) and "use cases" (functions).

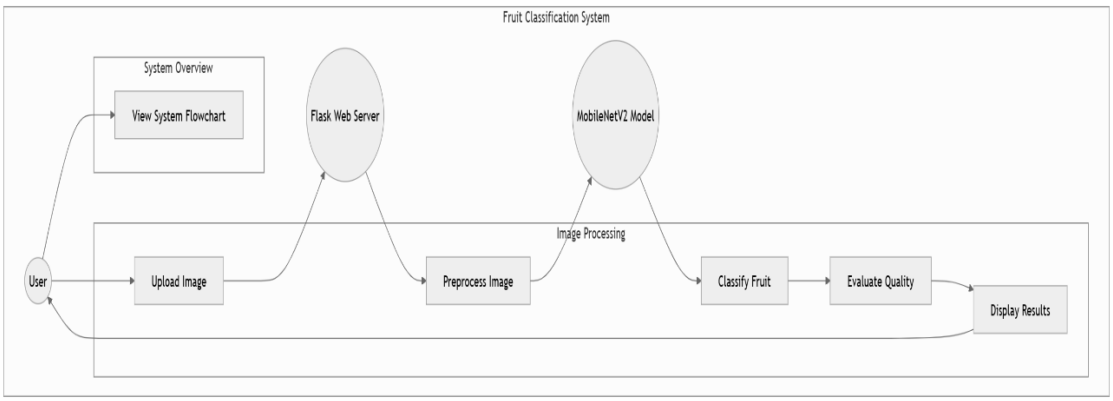

Fig 5.3.1: Use Case Diagram for Fruit Classification and Quality Evaluation

5.3.2. Class Diagram

A Class Diagram **shows the** static structure **of a system by representing** [22][24] its classes, attributes, methods, and relationships**.**

- Defines system components and their relationships.
- Uses concepts like inheritance, aggregation, and composition.

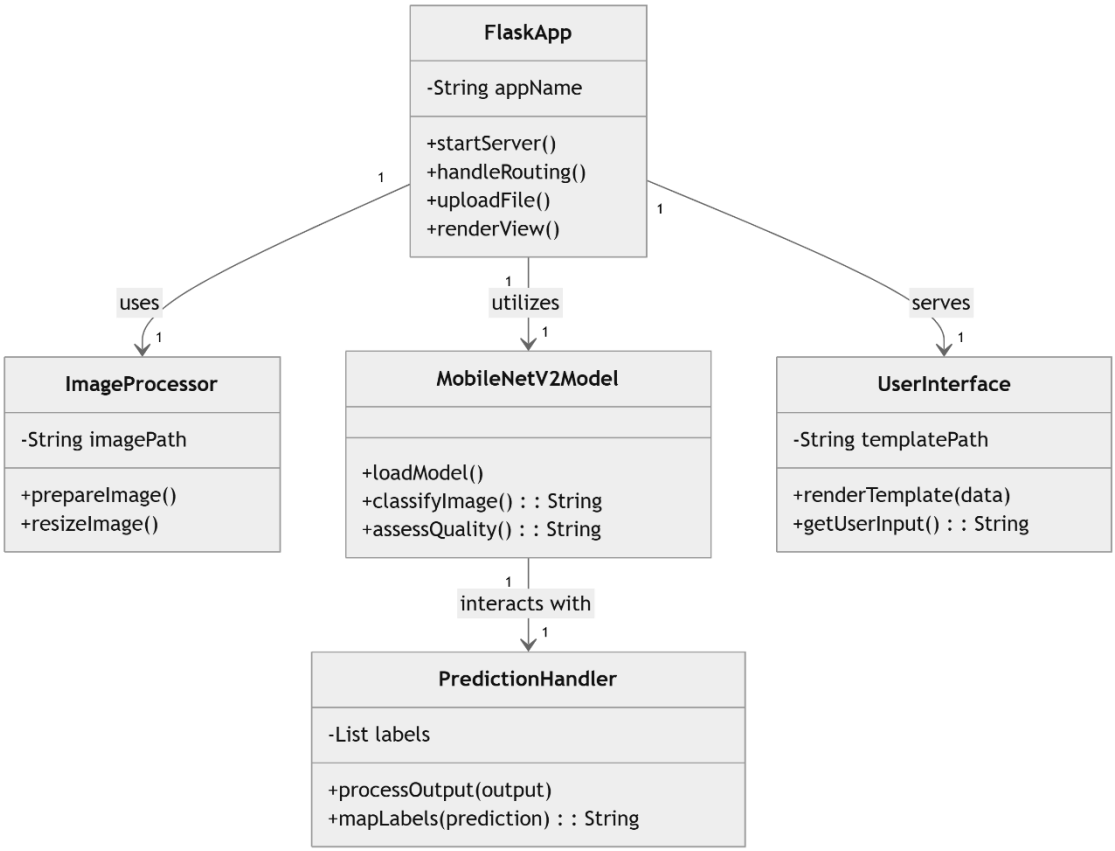

Fig 5.3.2 Class Diagram for Fruit Classification and Quality Evaluation

5.3.3. Activity Diagram

An **Activity Diagram** represents the **workflow of a system**, including conditions, decisions, and parallel processes[18][24].

- Represents parallel and conditional workflows.
- Helps in understanding the dynamic behavior of the system.

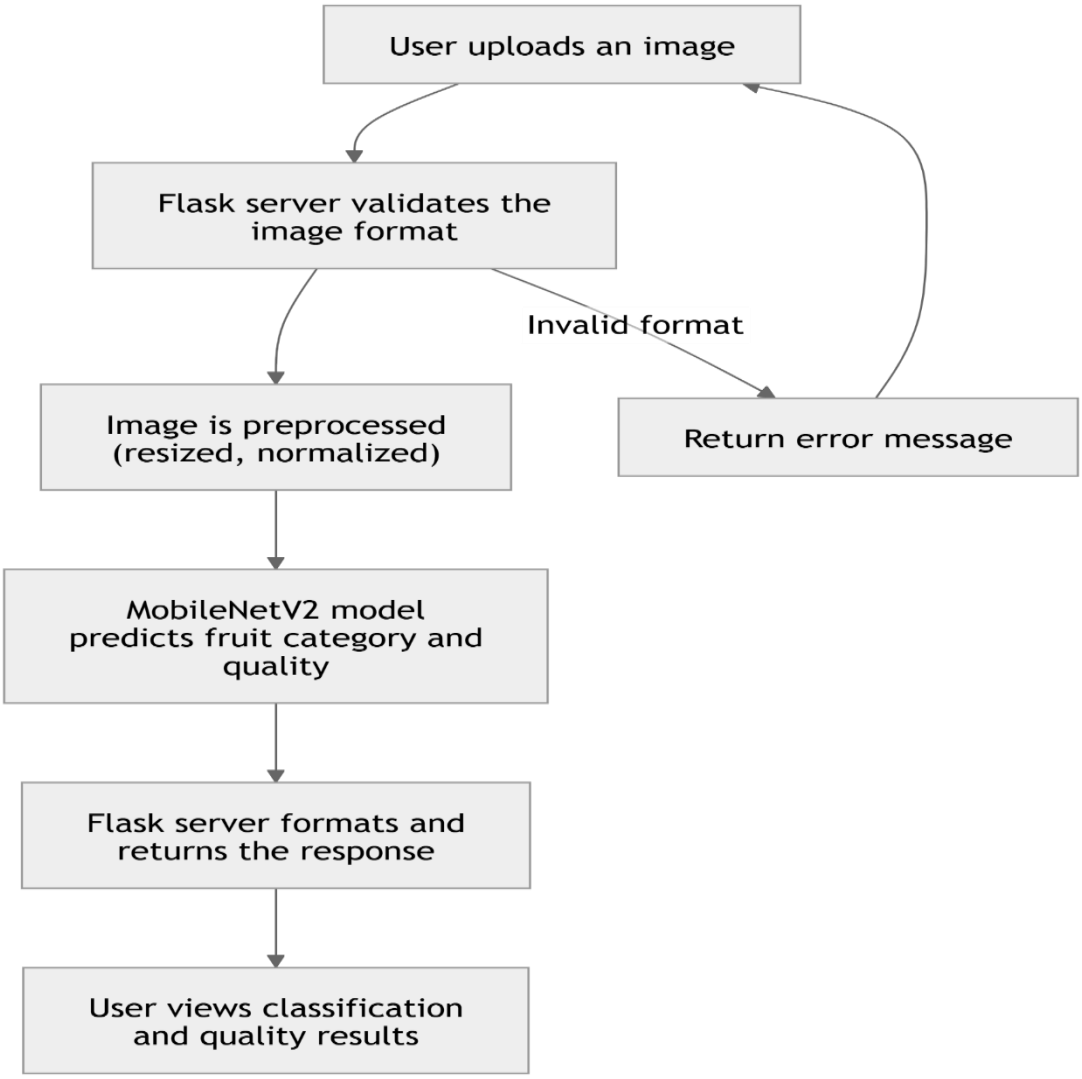

Fig 5.3.3 Activity Diagram for Fruit Classification and Quality Evaluation

5.3.4 Sequence Diagram

A Sequence Diagram **shows the** flow of interactions **between objects in a time-ordered sequence** [6][23].

- Focuses on the sequence of messages exchanged.
- Shows objects, lifelines, and interactions.
- Helps in understanding process execution.

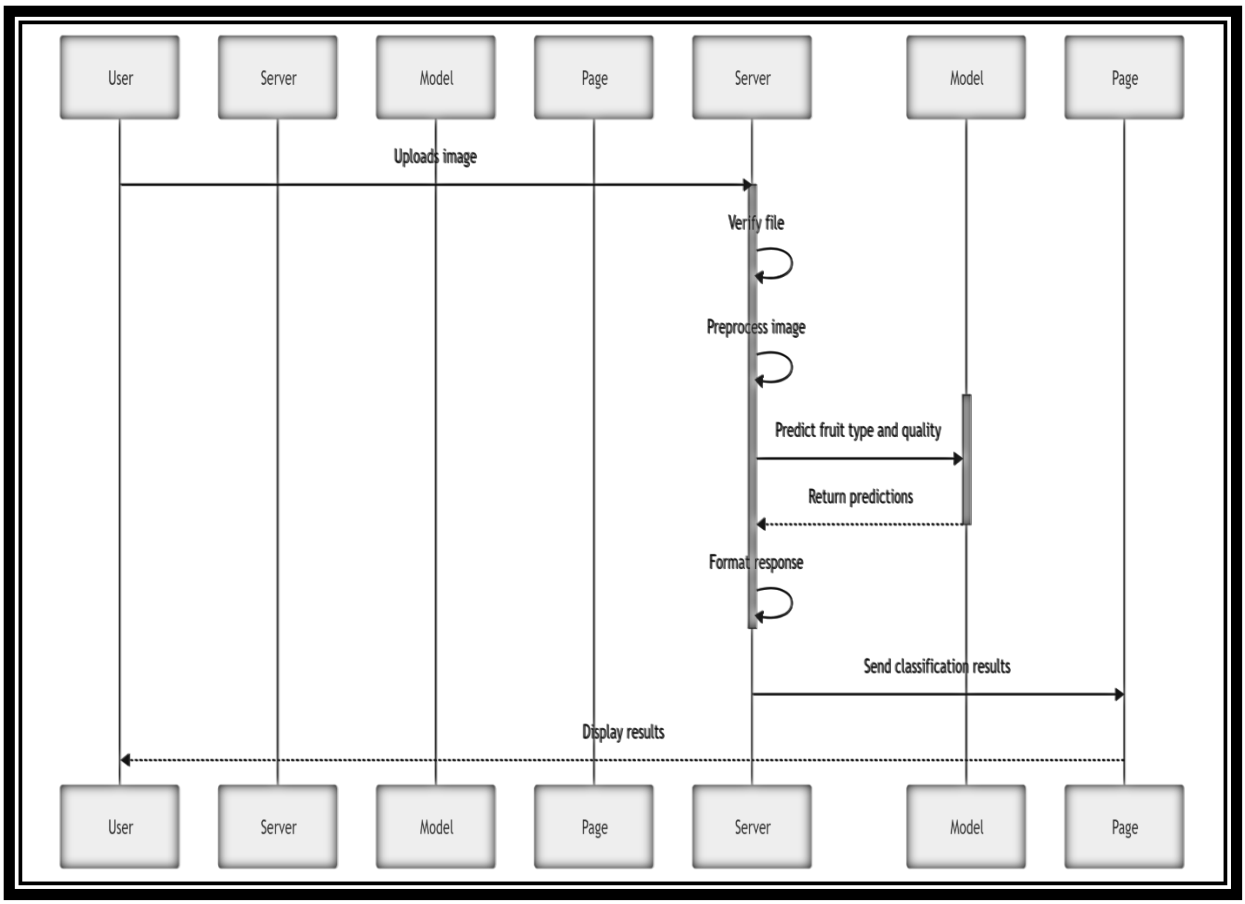

Fig 5.3.4 Sequential Diagram for Fruit Classification and Quality Evaluation

5.3.5 Interaction Diagram

An **Interaction Diagram** focuses on how objects interact with each other, showing message exchanges and collaborations.

- Emphasizes communication between components.
- Can be sequence, collaboration, or communication diagrams.
- Helps in system behavior analysis.

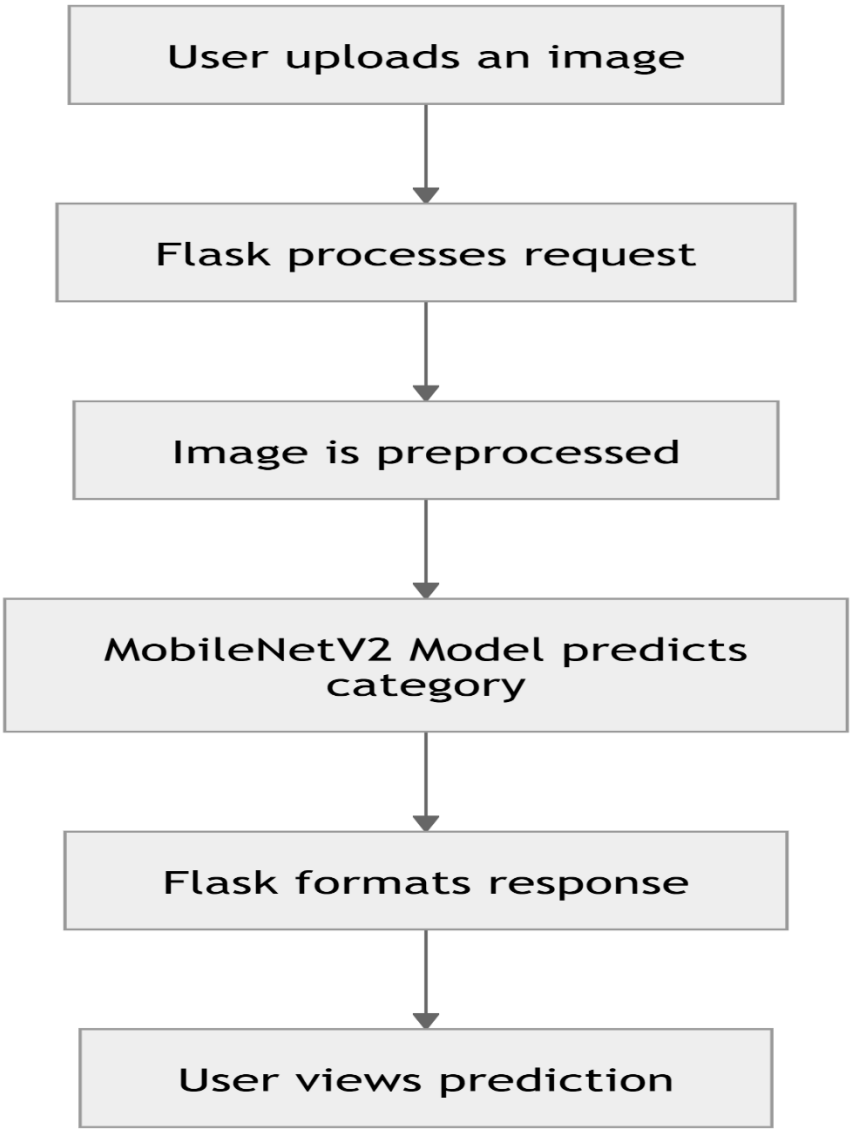

Fig 5.3.5 Interaction Diagram for Fruit Classification and Quality Evaluation

5.3.6 Deployment Diagram

A **Deployment Diagram** shows the **physical architecture** [4][7] of a system, including servers, software, and network configurations.

- Represents system hardware and software components.
- Shows how components interact physically.
- Helps in system deployment planning.

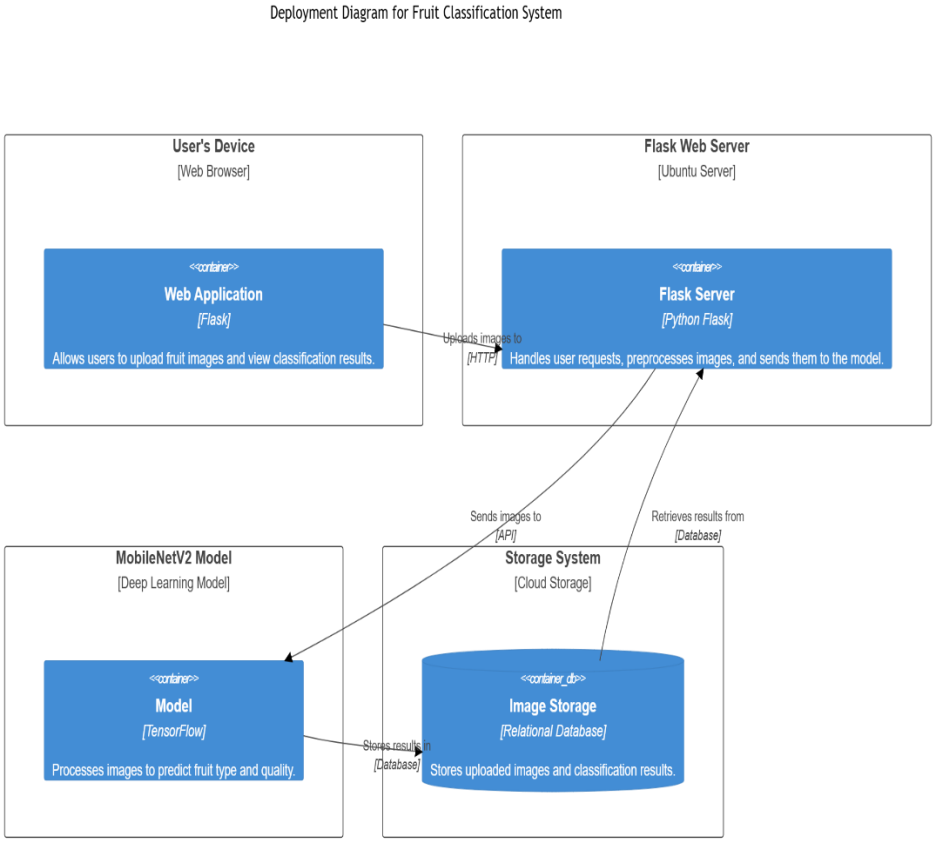

Fig 5.3.6 Deployment Diagram for Fruit Classification and Quality Evaluation

6.IMPLEMENTATION

6.1 Model Implementation

```
# prompt: summarize the mobilenetv2 model  like keras_cv.models.ImageClassifier.from_preset

# Assuming you want to summarize a MobileNetV2 model similar to how keras_cv does

# We'll use the standard Keras way to display model summary

# Load the MobileNetV2 model without the top (fully connected) layers

base_model = tf.keras.applications.MobileNetV2(weights='imagenet', include_top=False, input_shape=(128, 128, 3))

# Create a new model on top of the base model

model = Sequential()

model.add(base_model)

model.add(tf.keras.layers.GlobalAveragePooling2D())

model.add(Dense(256, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(num_classes, activation='softmax'))  # num_classes is from previous code

# Display the model summary

model.summary()

# prompt: train the model using MobileNet to get high accuracy

# Load the MobileNetV2 model without the top (fully connected) layers

base_model = tf.keras.applications.MobileNetV2(weights='imagenet', include_top=False, input_shape=(128, 128, 3))

# Freeze the base model layers

for layer in base_model.layers:

    layer.trainable = False

# Create a new model on top of the base model

model = Sequential()

model.add(base_model)

model.add(tf.keras.layers.GlobalAveragePooling2D())
```

```
model.add(Dense(256, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(num_classes, activation='softmax'))  # num_classes is from previous code

# Compile the model

model.compile(optimizer='adam',

              loss='categorical_crossentropy',

              metrics=['accuracy'])

# Train the model

history = model.fit(X_train, y_train_encoded, epochs=10, batch_size=32, validation_data=(X_test, y_test_encoded))

# Evaluate the model

loss, accuracy = model.evaluate(X_test, y_test_encoded)

print(f'Test Accuracy: {accuracy * 100:.2f}%')

# Save the entire model to a HDF5 file

model.save("my_mobilenet_model.h5")
```

6.2 Coding

Mount Google drive then Run

```
from google.colab import drive
drive.mount('/content/drive')
```

Unzip the Fruits Folder

```
!unzip -q '/content/drive/MyDrive/Fruit.zip' -d ata
```

Install keras packages

```
!pip install --upgrade --quiet keras-cv
!pip install --upgrade --quiet keras
```

Importing necessary modules

```
import os
os.environ["KERAS_BACKEND"] = "jax"
import keras
import keras_cv
import numpy as np
import tensorflow_datasets as tfds
import matplotlib
import tensorflow as tf
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Dense, Dropout
import tensorflow_hub as hub
from keras.layers import TFSMLayer
from keras.models import Sequential
from keras.layers import Dropout, Dense
```

Data Preprocessing

```
# Data Set taken total 18 classes with index labelling, which includes bad, good, mixed had 6 types fruits (total 18)

import numpy as np
import tensorflow as tf
import tensorflow_hub as hub
from sklearn.model_selection import train_test_split
import os
import cv2

# Example path to your image directory
image_dir = '/content/ata/Processed Images_Fruits'

# List to store resized images and labels
resized_images = []
labels = []
# Target size for resizing
target_size = (128, 128) # Better to take (64,64) Because of Memory Constraints

# Mapping for labels
label_to_index = {}
index = 0

# Iterate through each quality category
for quality in ['Bad Quality_Fruits', 'Good Quality_Fruits', 'Mixed Qualit_Fruits']:
    fruit_list = os.listdir(os.path.join(image_dir, quality))
    for fruit in fruit_list:
        fruit_path = os.path.join(image_dir, quality, fruit)
        img_list = os.listdir(fruit_path)
        for img_name in img_list:
            img_path = os.path.join(fruit_path, img_name)

            # Load and resize image
            img = cv2.imread(img_path)
              img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  # Convert BGR to RGB (OpenCV loads images in BGR format)
            resized_img = cv2.resize(img, target_size)

            # Append resized image and label
            resized_images.append(resized_img)
            if quality == 'Mixed Qualit_Fruits':
                label = fruit.lower() + '_mixed'
            else:
```

```
            label = fruit.lower()

        # Add label to mapping if not already present
        if label not in label_to_index:
            label_to_index[label] = index
            index += 1

        labels.append(label)

# Convert lists to numpy arrays
Image_data_resized = np.array(resized_images)
labels = np.array(labels)

# Convert string labels to integer labels
y = np.array([label_to_index[label] for label in labels])

# Check the shape of the resized array and labels
print("Resized Image Data Shape:", Image_data_resized.shape)
print("Labels Shape:", y.shape)

# Split data into training and testing sets
X_train,X_test,y_train,y_test=train_test_split(Image_data_resized,y,test_size=0.2,
random_state=42)
# Convert labels to categorical one-hot encoding
num_classes = len(label_to_index)
y_train_encoded=tf.keras.utils.to_categorical(y_train,num_classes=num_classe)
y_test_encoded = tf.keras.utils.to_categorical(y_test, num_classes=num_classes)

# Normalize pixel values to the range [0, 1]
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0
```

Display any 2 images after preprocessing from training set

```
# prompt: display any 2 images after preprocessing from training set

import numpy as np
import matplotlib.pyplot as plt
# Select 2 random indices from the training set
indices = np.random.choice(X_train.shape[0], 2, replace=False)
# Display the images
plt.figure(figsize=(10, 5))
for i, idx in enumerate(indices):
  plt.subplot(1, 2, i + 1)
  plt.imshow(X_train[idx])
  plt.title(f"Label: {y_train[idx]}")
  plt.axis('off')
plt.show()
```

Display any 2 images after preprocessing from testing set

```
# prompt: display any 2 images after preprocessing from testing set
```

```
import matplotlib.pyplot as plt
import numpy as np
# Select 2 random indices from the testing set
indices = np.random.choice(X_test.shape[0], 2, replace=False)
# Display the images
plt.figure(figsize=(10, 5))
for i, idx in enumerate(indices):
  plt.subplot(1, 2, i + 1)
  plt.imshow(X_test[idx])
  plt.title(f"Label: {y_test[idx]}")
  plt.axis('off')
  plt.show()
```

Model.summmary() for EfficientNetv2:-

```
model = keras_cv.models.ImageClassifier.from_preset(
    "efficientnetv2_b0_imagenet",
    num_classes=num_classes,
)
model.summary()


# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Print model s
```

Train the model using EfficientNetv2 and save it

```
# Train the model
history = model.fit(X_train, y_train_encoded, epochs=10, batch_size=32,
validation_data=(X_test, y_test_encoded))

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test_encoded)
print(f'Test Accuracy: {accuracy * 100:.2f}%')

# Save the entire model to a HDF5 file
model.save("my_model.h5")
```

model summary on vgg16 model

```
# prompt: model summary on vgg16 model

from tensorflow.keras.applications import VGG16

# Load pre-trained VGG16 model (excluding top classification layers)
base_model = VGG16(weights='imagenet', include_top=False,
input_shape=(target_size[0], target_size[1], 3))
```

```
# Freeze the base model layers
for layer in base_model.layers:
  layer.trainable = False

# Create a new model on top
model = Sequential()
model.add(base_model)
model.add(tf.keras.layers.GlobalAveragePooling2D())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Print model summary
model.summary()
```

train the model using vgg16

```
# prompt: train the above using vgg16

from tensorflow.keras.applications import VGG16

# Load the VGG16 model without the top (fully connected) layers
base_model=VGG16(weights='imagenet',include_top=False,  input_shape=(128,  128,
3))
# Freeze the base model layers
for layer in base_model.layers:
    layer.trainable = False

# Create a new model on top of the base model
model = Sequential()
model.add(base_model)
model.add(tf.keras.layers.GlobalAveragePooling2D())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))  # num_classes is from previous
code

# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
```

```
history = model.fit(X_train, y_train_encoded, epochs=10, batch_size=32, validation_data=(X_test, y_test_encoded))

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test_encoded)
print(f'Test Accuracy: {accuracy * 100:.2f}%')

# Save the entire model to a HDF5 file
model.save("my_vgg16_model.h5")
```

model summary on inceptionv3 model

```
# prompt: model summary on inceptionv3 model
from tensorflow.keras.applications import InceptionV3
base_model=InceptionV3(weights='imagenet',include_top=False,input_shape=(target_size[0], target_size[1], 3))

# Freeze the base model layers
for layer in base_model.layers:
    layer.trainable = False

# Create a new model on top of the base model
model = Sequential()
model.add(base_model)
model.add(tf.keras.layers.GlobalAveragePooling2D())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))  # num_classes is from previous code
# Compile the model (optional for just getting the summary)
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
# Print model summary
model.summary()
```

Train inception v3 model

```
# prompt: train inception v3 model
from tensorflow.keras.applications import InceptionV3
# Load the InceptionV3 model without the top (fully connected) layers
base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=(128, 128, 3))
# Freeze the base model layers
for layer in base_model.layers:
    layer.trainable = False
# Create a new model on top of the base model
model = Sequential()
model.add(base_model)
model.add(tf.keras.layers.GlobalAveragePooling2D())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
```

```
model.add(Dense(num_classes, activation='softmax'))  # num_classes is from previous code

# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train_encoded, epochs=10, batch_size=32, validation_data=(X_test, y_test_encoded))

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test_encoded)
print(f'Test Accuracy: {accuracy * 100:.2f}%')

# Save the entire model to a HDF5 file
model.save("my_model.keras.h5")
```

MODEL SUMMARY ON DENSENET

```
# prompt: MODEL SUMMARY ON DENSENET
from tensorflow.keras.applications import DenseNet121
base_model=DenseNet121(weights='imagenet',include_top=False,
input_shape=(target_size[0], target_size[1], 3))
# Freeze the base model layers (optional, depending on your fine-tuning strategy)
for layer in base_model.layers:
    layer.trainable = False
# Create a new model on top of the base model
model = Sequential()
model.add(base_model)
model.add(tf.keras.layers.GlobalAveragePooling2D())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

# Compile the model (optional for just getting the summary)
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy']
)
# Print model summary
model.summary()
```

Train the above model using densenet

```
# prompt: train the above model using densenet
from tensorflow.keras.applications import DenseNet121
# Load the DenseNet121 model without the top (fully connected) layers
base_model=DenseNet121(weights='imagenet',include_top=False,input_shape=(128,
128, 3))
# Freeze the base model layers
for layer in base_model.layers:
    layer.trainable = False
```

```
# Create a new model on top of the base model
model = Sequential()
model.add(base_model)
model.add(tf.keras.layers.GlobalAveragePooling2D())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))  # num_classes is from previous code

# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train_encoded, epochs=10, batch_size=32, validation_data=(X_test, y_test_encoded))

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test_encoded)
print(f'Test Accuracy: {accuracy * 100:.2f}%')

# Save the entire model to a HDF5 file
model.save("my_densenet_model.h5")
```

MODEL SUMMARY ON NASNET

```
# prompt: MODEL SUMMARY ON NASNET
base_model = tf.keras.applications.NASNetMobile(
    input_shape=(target_size[0], target_size[1], 3),
    include_top=False,
    weights='imagenet'
)
# Freeze the base model layers (optional, depending on your fine-tuning strategy)
for layer in base_model.layers:
    layer.trainable = False

# Create a new model on top of the base model
model = Sequential()
model.add(base_model)
model.add(tf.keras.layers.GlobalAveragePooling2D())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

# Compile the model (optional for just getting the summary)
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy']
)
# Print model summary
model.summary()
```

Train the model using nasnet

```
# prompt: Train the model using nasnet
# Load the NASNetMobile model without the top (fully connected) layers
base_model=tf.keras.applications.NASNetMobile(weights='imagenet',
include_top=False, input_shape=(128, 128, 3))
# Freeze the base model layers
for layer in base_model.layers:
   layer.trainable = False
# Create a new model on top of the base model
model = Sequential()
model.add(base_model)
model.add(tf.keras.layers.GlobalAveragePooling2D())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))  # num_classes is from previous
code
# Compile the model
model.compile(optimizer='adam',
            loss='categorical_crossentropy',
            metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train_encoded, epochs=10, batch_size=32,
validation_data=(X_test, y_test_encoded))

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test_encoded)
print(f'Test Accuracy: {accuracy * 100:.2f}%')

# Save the entire model to a HDF5 file
model.save("my_nasnet_model.h5")
```

MODEL SUMMARY ON XCEPTION

```
# prompt: MODEL SUMMARY ON XCEPTION

base_model = tf.keras.applications.Xception(
   input_shape=(target_size[0], target_size[1], 3),
   include_top=False,
   weights='imagenet'
)

# Freeze the base model layers (optional, depending on your fine-tuning strategy)
for layer in base_model.layers:
   layer.trainable = False

# Create a new model on top of the base model
model = Sequential()
model.add(base_model)
model.add(tf.keras.layers.GlobalAveragePooling2D())
```

```
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

# Compile the model (optional for just getting the summary)
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy']
)

# Print model summary
model.summary()
```

Train the model using xception

```
# prompt: Train the model using xception

# Load the Xception model without the top (fully connected) layers
base_model=tf.keras.applications.Xception(weights='imagenet',     include_top=False,
input_shape=(128, 128, 3))

# Freeze the base model layers
for layer in base_model.layers:
    layer.trainable = False

# Create a new model on top of the base model
model = Sequential()
model.add(base_model)
model.add(tf.keras.layers.GlobalAveragePooling2D())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))  # num_classes is from previous
code

# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train_encoded, epochs=10, batch_size=32,
validation_data=(X_test, y_test_encoded))

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test_encoded)
print(f'Test Accuracy: {accuracy * 100:.2f}%')

# Save the entire model to a HDF5 file
model.save("my_xception_model.h5")
```

Classification report for mobilenetv2 model:-

```
import numpy as np
from sklearn.metrics import classification_report
```

```
# Predict classes for test data
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
# Generate classification report
report=classification_report(y_test,y_pred_classes,target_names=label_to_index.keys
())
print(report)
```

cross entropy for mobilenet

```
import matplotlib.pyplot as plt
# Extract loss values from the training history
train_loss = history.history['loss']
val_loss = history.history['val_loss']
# Create a plot for loss
plt.figure(figsize=(10, 6))
plt.plot(train_loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Cross-entropy Loss')
plt.title('Training and Validation Loss (MobileNet)')
plt.legend()
plt.grid(True)
plt.show()
```

Accuracies of mobilenet using graph

```
import matplotlib.pyplot as plt
# Extract accuracy values from the training history
train_accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']

# Create a plot for accuracy
plt.figure(figsize=(10, 6))
plt.plot(train_accuracy, label='Training Accuracy')
plt.plot(val_accuracy, label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy (MobileNet)')
plt.legend()
plt.grid(True)
plt.show()
```

MobileNetV2 Accuracy over Epochs on T4 GPU:-

```
import matplotlib.pyplot as plt
# Data for the graph
epochs = list(range(1, 11))
train_accuracy = [0.8406, 0.9736, 0.9783, 0.9852, 0.9933, 0.9955, 0.9872, 0.9928,
0.9947, 0.9956]
val_accuracy = [0.9747, 0.9854, 0.9928, 0.9857, 0.9916, 0.9903, 0.9898, 0.9956,
0.9910, 0.9900]
# Plotting the graph
```

```
plt.figure(figsize=(10, 6))
plt.plot(epochs, train_accuracy, label="Training Accuracy", marker='o')
plt.plot(epochs, val_accuracy, label="Validation Accuracy", marker='s')
plt.title('MobileNetV2 Accuracy over Epochs on T4 GPU')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.ylim([0.8, 1.0])
plt.grid(True)
plt.legend(loc='lower right')
plt.tight_layout()
plt.show()
```

Flask

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
     <title>DeepFruit: A Unified Deep Learning Framework for Fruit Classification and  &amp;nbsp; &amp;nbsp; Quality Evaluation
</title>
<link
rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
   <style>
      body {
         font-family: Arial, sans-serif;
         background-color: #f5f5f5;

      }
      .navbar {
         margin-bottom: 0;
         border-radius: 0;
         background-color: #2c6975;
         color: white;
         padding: 10px 15px;

      }

      .navbar-brand, .navbar-nav li a {
         color: white !important;
      }

      .navbar-brand:hover, .navbar-nav li a:hover {
         color: #d4f1f4 !important;
      }

      .navbar-nav {
```

```
    display: flex;
    justify-content: space-around;
    width: 100%;
}
.title-section {
    display: flex;
    align-items: center;
    justify-content: center;
    padding: 2px;
    background-color: #eafaf1;
    color: #2c3e50;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

.title-section img {
    width: 120px;
    height: auto;
    margin-right: 20px;
}

.title-text {
    text-align: left;
}

.title-text h1 {
    font-size: 2.5em;
    font-weight: bold;
    margin: 0;
}

.title-text h3 {
    font-size: 1.2em;
    font-weight: 400;
    color: #34495e;
    margin-top: 10px;
}

.container {
    margin-top: 20px;
}

.form-inline {
    margin-top: 30px;
    text-align: center;
}

input[type="file"] {
    margin: 10px 0;
}
```

```
        .btn-success {
            background-color: #2c6975;
            border: none;
            transition: background-color 0.3s ease;
        }

        .btn-success:hover {
            background-color: #8fc1b5;
        }


        footer {
            font-weight: bold;
            font-family: Verdana, Geneva, Tahoma, sans-serif;
            text-align: center;
            padding: 20px;
            background: #2c6975;
            color: #f4f9f4;
            font-size: 1.5rem;
            margin-top: 0px;
        }

        footer a {
            color: #8fc1b5;
            text-decoration: none;
        }

        footer a:hover {
            text-decoration: underline;
        }

    </style>
</head>
<body>
    <div class="title-section">
        <img src="/static/images/logo.jpg" alt="Logo">
        <div class="title-text">
            <h1>DeepFruit: A Unified Deep Learning Framework for Fruit Classification
<br>
             and Quality Evaluation</h1>
             <h3><b>Team Members:-</b> Bollineni Yasaswini &bull; Aravapalli Vasavi
&bull;Nakka Annapurna 
              Guide:-Dr.S.N.Tirumala Rao
Mentor:-Dodda VenkatReddy</h3>
        </div>
    </div>

    <nav class="navbar navbar-inverse">
        <div class="container-fluid">
            <div class="navbar-header">
```

```
            </div>
            <ul class="nav navbar-nav">
                <li><a href="/">Home</a></li>
                <li><a href="/about">About</a></li>
                <li><a href="/predict">Predict</a></li>
                <li><a href="/model">Model Evaluation</a></li>
                <li><a href="/flowchart">Flowchart</a></li>


            </ul>
        </div>
    </nav>
    <img src="/static/images/fr.jpg" height="600px" width="1350px">
</body>
<footer>
        &copy; 2024 Fruit Quality Classifier | Yasaswini 21471A0512 |
yasaswinibollineni25@gmail.com
</footer>
</html>
```

About.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>About - Fruit Quality Classifier</title>
   <link
rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #2c6975;
            margin: 0;
            padding: 0px;
        }

        .navbar {
            margin-bottom: 0;
            border-radius: 0;
            background-color: #2c6975;
            color: white;
            padding: 10px 15px;
        }

        .navbar-brand, .navbar-nav li a {
            color: white !important;
        }
```

```
.navbar-brand:hover, .navbar-nav li a:hover {
    color: #d4f1f4 !important;
}

.navbar-nav {
    display: flex;
    justify-content: space-around;
    width: 100%;
}

.container {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    align-items: center;
    margin: 50px auto;
    background: #fff;
    border-radius: 10px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    overflow: hidden;

}

.content {
    flex: 1;
    padding: 40px;
}

.content h1 {
    color: #2c6975;
}

.content p {
    font-size: 18px;
    line-height: 1.6;
    color: #333;
}

.image-section {
    flex: 1;
    background-color: #ffe066;
    display: flex;
    justify-content: center;
    align-items: center;
    padding-top: 0px;
    height:500px ;
}

.image-section img {
    max-width: 100%;
```

```
            height: auto;
            height: 500px;
        }

        footer {
            font-weight: bold;
            font-family: Verdana, Geneva, Tahoma, sans-serif;
            text-align: center;
            padding: 20px;
            background: #2c6975;
            color: #f4f9f4;
            font-size: 1.5rem;
            margin-top: 20px;
        }

        footer a {
            color: #8fc1b5;
            text-decoration: none;
        }

        footer a:hover {
            text-decoration: underline;
        }
    </style>
</head>
<body>
    <nav class="navbar navbar-inverse">
        <div class="container-fluid">
            <ul class="nav navbar-nav">
                <li><a href="/">Home</a></li>
                <li><a href="/about">About</a></li>
                <li><a href="/predict">Predict</a></li>
                <li><a href="/model">Model Evaluation</a></li>
                <li><a href="/flowchart">Flowchart</a></li>
            </ul>
        </div>
    </nav>

    <div class="container">
        <!-- Left Content -->
        <div class="content">
            <h1>About the Project</h1>
            <p>
                Our project focuses on automating fruit classification and grading using deep
learning,
                specifically with the MobileNetV2 architecture. It processes images of fruits
to classify
                them by type and assess their quality accurately. This solution is essential for
agriculture
```

```
            as it streamlines the sorting and grading process, reducing manual labor and
human error.
          </p>
          <p>
             By implementing this technology, farmers and distributors can improve the
efficiency and
             consistency of fruit quality checks. It also enhances market transparency by
providing
              objective quality assessments. The use of transfer learning and computer
vision ensures high
             accuracy while keeping computational costs manageable.
          </p>
          <p>
            This innovation supports sustainable farming practices by minimizing waste.
Ultimately,
             it bridges traditional agricultural methods with modern technology, boosting
productivity
             and profitability.
          </p>
       </div>
       <!-- Right Image -->
       <div class="image-section">
          <img src="/static/images/about.jpg" alt="Fruit Illustration">
       </div>
    </div>

    <footer>
                &copy; 2024 Fruit Quality Classifier | Yasaswini 21471A0512 |
yasaswinibollineni25@gmail.com
    </footer>
</body>
</html>
```

Predict.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Yasaswini -21471A0512 Fruit Quality Classifier </title>
                                                                                <link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;500&family=Lor
a:wght@400;700&display=swap" rel="stylesheet">
                                                          <link          rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <style>
        body {
            font-family: 'Roboto', sans-serif;
            margin: 0;
            padding: 0;
```

```
    background: linear-gradient(to bottom, #eafaf1, #d4f1f4);
    color: #2c3e50;
}

header {
    text-align: center;
    padding: 20px 0;
    background: #2c6975;
    color: #f4f9f4;
    font-family: 'Lora', serif;
    font-size: 3rem;
    letter-spacing: 2px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
}

.container {
    margin: 50px auto;
    padding: 40px;
    max-width: 800px;
    background: linear-gradient(to bottom right, #ffffff, #f1c40f);
    border-radius: 15px;
    box-shadow: 0 10px 20px rgba(0, 0, 0, 0.3);
}

h1 {
    font-size: 2.5rem;
    color: #e74c3c;
    text-shadow: 2px 2px 4px #000000;
    margin-bottom: 20px;
}

p {
    font-size: 1.2rem;
    color: #34495e;
    margin-bottom: 25px;
}

form {
    margin-top: 20px;
}

input[type="file"] {
    padding: 10px;
    background: #f4f9f4;
    border: 2px solid #2980b9;
    border-radius: 8px;
    font-size: 1rem;
    margin-bottom: 20px;
    outline: none;
    color: #2c3e50;
```

```
    transition: 0.3s;
}

input[type="file"]:hover {
    border-color: #2c6975;
}

button {
    background-color: #8fc1b5;
    color: #ffffff;
    border: none;
    padding: 15px 40px;
    font-size: 1.5rem;
    font-weight: bold;
    border-radius: 20px;
    cursor: pointer;
    transition: all 0.3s;
    outline: none;
}

button:hover {
    background-color: #2c6975;
    transform: scale(1.1);
    box-shadow: 0 8px 15px rgba(44, 105, 117, 0.4);
}

.row {
    margin-top: 30px;
}

.img-thumbnail {
    border: 4px solid #e67e22;
    border-radius: 60%;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    width: 250px;
    height: 250px;
}

.info-container h4 {
    margin-bottom: 15px;
    font-size: 1.4em;
    font-weight: bold;
    text-shadow: 1px 1px 2px #000000;
}

.info-container mark {
    padding: 8px 12px;
    font-size: 1.2em;
    background: linear-gradient(90deg, #f39c12, #d35400);
    color: white;
```

```
            border-radius: 8px;
            text-shadow: 1px 1px 2px #000000;
        }

        footer {
            font-weight: bold;
            font-family: Verdana, Geneva, Tahoma, sans-serif;
            text-align: center;
            padding: 20px;
            background: #2c6975;
            color: #f4f9f4;
            font-size: 1.5rem;
            margin-top: 50px;
        }

        footer a {
            color: #8fc1b5;
            text-decoration: none;
        }

        footer a:hover {
            text-decoration: underline;
        }

        @media (max-width: 768px) {
            .col-lg-6 {
                text-align: center;
                margin-bottom: 20px;
            }

            .img-thumbnail {
                float: none;
                margin: 0 auto;
                width: 200px;
                height: 200px;
            }
        }
    </style>
</head>
<body>
    <header>
        Fruit Quality Classifier
        <a href="/" style="color:whitesmoke;">Go Back</a>
    </header>
    <div class="container">
        <h1>Upload Your Fruit</h1>
        <p>Find out the type and quality of your fruit.</p>
                        <form id="uploadForm" action="/predict" method="post"
enctype="multipart/form-data">
            <input type="file" name="file" accept="image/*" required>
```

```
            <button type="submit">Predict</button>
        </form>
        <div class="row">
            <div class="col-lg-6 text-center">
                <img src="{{ user_image }}" alt="User Image" class="img-thumbnail">
            </div>
            <div class="col-lg-6 info-container">
                <h4>Fruit Name: <mark>{{ fruit }}</mark></h4>
                <h4>Fruit Quality: <mark>{{ quality }}</mark></h4>
            </div>
        </div>
    </div>
    <footer>
                &copy; 2024 Fruit Quality Classifier | Yasaswini 21471A0512 |
yasaswinibollineni25@gmail.com
    </footer>
</body>
</html>
```

Model.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <link
rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f5f5f5;
        }

        .navbar {
            margin-bottom: 0;
            border-radius: 0;
            background-color: #2c6975;
            color: white;
            padding: 10px 15px;
        }

        .navbar-brand, .navbar-nav li a {
            color: white !important;
        }

        .navbar-brand:hover, .navbar-nav li a:hover {
            color: #d4f1f4 !important;
        }
```

```
.navbar-nav {
    display: flex;
    justify-content: space-around;
    width: 100%;
}

footer {
    font-weight: bold;
    font-family: Verdana, Geneva, Tahoma, sans-serif;
    text-align: center;
    padding: 20px;
    background: #2c6975;
    color: #f4f9f4;
    font-size: 1.5rem;
    margin-top: 0px;
}

footer a {
    color: #8fc1b5;
    text-decoration: none;
}

footer a:hover {
    text-decoration: underline;
}

.table-container {
    margin: 20px auto;
    width: 90%;
    background-color: white;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

table {
    width: 100%;
    border-collapse: collapse;
    margin: 20px 0;
}

th, td {
    text-align: center;
    padding: 8px;
    border: 1px solid #ddd;
}

th {
    background-color: #2c6975;
    color: white;
```

```
        }
    </style>
</head>
<body>
    <nav class="navbar navbar-inverse">
        <div class="container-fluid">
            <ul class="nav navbar-nav">
                <li><a href="/">Home</a></li>
                <li><a href="/about">About</a></li>
                <li><a href="/predict">Predict</a></li>
                <li><a href="/model">Model Evaluation</a></li>
                <li><a href="/flowchart">Flowchart</a></li>
            </ul>
        </div>
    </nav>

    <div class="table-container">
        <h3>Fruit Quality Evaluation</h3>
        <table>
            <thead>
                <tr>
                    <th>Model</th>
                    <th>Accuracy</th>
                    <th>Precision</th>
                    <th>Recall</th>
                    <th>F1-Score</th>

                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>MobileNetV2</td>
                    <td>99.7%</td>
                    <td>0.98</td>
                    <td>0.98</td>
                    <td>0.97</td>
                </tr>

                <tr>
                    <td>EfficientNetV2</td>
                    <td>99.5%</td>
                    <td>0.98</td>
                    <td>0.95</td>
                    <td>0.96</td>

                </tr>
                <tr>
                    <td>VGG16</td>
                    <td>96%</td>
                    <td>0.98</td>
```

```
            <td>0.92</td>
            <td>0.91</td>

        </tr>
        <tr>
            <td>InceptionV3</td>
            <td>95.75%</td>
            <td>0.95</td>
            <td>0.98</td>
            <td>0.97</td>

        </tr>
        <tr>
            <td>DenseNet</td>
            <td>98%</td>
            <td>0.97</td>
            <td>0.97</td>
            <td>0.97</td>

        </tr>
        <tr>
            <td>NASNet</td>
            <td>97.6%</td>
            <td>0.96</td>
            <td>0.98</td>
            <td>0.98</td>

        </tr>
        <tr>
            <td>Xception</td>
            <td>97%</td>
            <td>0.97</td>
            <td>0.97</td>
            <td>0.95</td>

        </tr>
    </tbody>
</table>

<h3>Fruit Type Evaluation</h3>
<table>
    <thead>
        <tr>
            <th>Model</th>
             <th>Accuracy</th>
            <th>Precision</th>
            <th>Recall</th>
            <th>F1-Score</th>

        </tr>
```

```
</thead>
<tbody>

   <tr>
      <td>MobileNetV2</td>
      <td>99.7%</td>
      <td>0.99</td>
      <td>0.98</td>
      <td>0.98</td>
   </tr>

   <tr>
      <td>EfficientNetV2</td>
      <td>99.5%</td>
      <td>0.99</td>
      <td>0.98</td>
      <td>0.97</td>
   </tr>
   <tr>
      <td>VGG16</td>
      <td>96%</td>
      <td>0.95</td>
      <td>0.93</td>
      <td>0.94</td>

   </tr>
   <tr>
      <td>InceptionV3</td>
      <td>95.75%</td>
      <td>0.96</td>
      <td>0.97</td>
      <td>0.95</td>

   </tr>
   <tr>
      <td>DenseNet</td>
      <td>98%</td>
      <td>0.98</td>
      <td>0.98</td>
      <td>0.96</td>

   </tr>
   <tr>
      <td>NASNet</td>
      <td>97.6%</td>
      <td>0.97</td>
      <td>0.98</td>
      <td>0.99</td>

   </tr>
```

```
            <tr>
              <td>Xception</td>
              <td>97%</td>
              <td>0.98</td>
              <td>0.99</td>
              <td>0.95</td>

            </tr>
          </tbody>
        </table>
    </div>

    <footer>
                &copy; 2024 Fruit Quality Classifier | Yasaswini 21471A0512 |
yasaswinibollineni25@gmail.com
    </footer>
</body>
</html>
```

Flowchart.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

 <link
rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f5f5f5;
        }

        .navbar {
            margin-bottom: 0;
            border-radius: 0;
            background-color: #2c6975;
            color: white;
            padding: 10px 15px;
        }

        .navbar-brand, .navbar-nav li a {
            color: white !important;
        }

        .navbar-brand:hover, .navbar-nav li a:hover {
            color: #d4f1f4 !important;
        }
```

```
.navbar-nav {
    display: flex;
    justify-content: space-around;
    width: 100%;
}

footer {
    font-weight: bold;
    font-family: Verdana, Geneva, Tahoma, sans-serif;
    text-align: center;
    padding: 20px;
    background: #2c6975;
    color: #f4f9f4;
    font-size: 1.5rem;
    margin-top: 0px;
}

footer a {
    color: #8fc1b5;
    text-decoration: none;
}

footer a:hover {
    text-decoration: underline;
}

body {
    background-color: #2c6975;
}

img {
    padding-left: 300px;
    padding-top: 5px;
    animation: flip 1s ease-in-out;
}

@keyframes flip {
    0% {
        transform: rotateY(0deg);
    }
    50% {
        transform: rotateY(90deg);
    }
    100% {
        transform: rotateY(0deg);
    }
}

/* Additional effects */
```

```
        .navbar-nav li a {
            transition: color 0.3s ease-in-out;
        }

        img:hover {
            transform: scale(1.05);
            transition: transform 0.3s ease-in-out;
        }
    </style>
</head>
<body>
    <nav class="navbar navbar-inverse">
        <div class="container-fluid">
            <div class="navbar-header">
            </div>
            <ul class="nav navbar-nav">
                <li><a href="/">Home</a></li>
                <li><a href="/about">About</a></li>
                <li><a href="/predict">Predict</a></li>
                <li><a href="/model">Model Evaluation</a></li>
                <li><a href="/flowchart">Flowchart</a></li>
            </ul>
        </div>
    </nav>

    <img src="/static/images/flowchart.png" height="530px" width="1050px">

</body>
<footer>
        &copy; 2024 Fruit Quality Classifier | Yasaswini 21471A0512 | yasaswinibollineni25@gmail.com
</footer>
</html>
```

App.py

```
from flask import Flask, render_template, request
from tensorflow.keras.models import load_model
import numpy as np
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
 #from tensorflow.keras.applications.vgg16 import preprocess_input

import os
from tensorflow.keras.preprocessing import image

app = Flask(__name__)
model = load_model('go.h5')
target_img = os.path.join(os.getcwd(), 'static/images')
```

```
@app.route('/')
def index_view():
    return render_template('index.html')
@app.route('/about')
def about_view():
    return render_template('about.html')
@app.route('/model')
def model_eval():
    return render_template('model.html')
@app.route('/flowchart')
def flowchart_view():
    return render_template('flowchart.html')
    # Allow files with extension png, jpg, and jpeg
ALLOWED_EXT = set(['jpg', 'jpeg', 'png'])

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1] in ALLOWED_EXT

# Function to load and prepare the image in the right shape
def read_image(filename):
    img = load_img(filename, target_size=(224, 224))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)
    return x

@app.route('/predict', methods=['GET', 'POST'])
def predict():
    if request.method == 'POST':
        file = request.files['file']
        if file and allowed_file(file.filename):
            filename = file.filename
            file_path = os.path.join('static/images', filename)
            file.save(file_path)
            img = read_image(file_path)
            class_prediction = model.predict(img)
            classes_x = np.argmax(class_prediction, axis=1)

            # Map the class to fruit name
            fruit = ""
            quality = ""

            if classes_x == 0:
                fruit = "Apple"
            elif classes_x == 1:
                fruit = "Banana"
            elif classes_x == 2:
                fruit = "Orange"
            elif classes_x == 3:
                fruit = "Lime"
```

```
        elif classes_x == 4:
            fruit = "Guava"
        elif classes_x == 5:
            fruit = "Pomogranate"

        # Assign quality based on probabilities
        max_prob = np.max(class_prediction)
        if max_prob > 0.8:
            quality = "Good"
        elif max_prob > 0.5:
            quality = "Mixed"
        else:
            quality = "Bad"

                    return render_template('predict.html', fruit=fruit, quality=quality,
user_image=file_path)
        else:
            return "Unable to read the file. Please check file extension."
    elif request.method == 'GET':
        # Render the predict.html form page
        return render_template('predict.html')

if __name__ == '__main__':
    app.run(debug=True, use_reloader=False, port=8000)
```

Execution:-

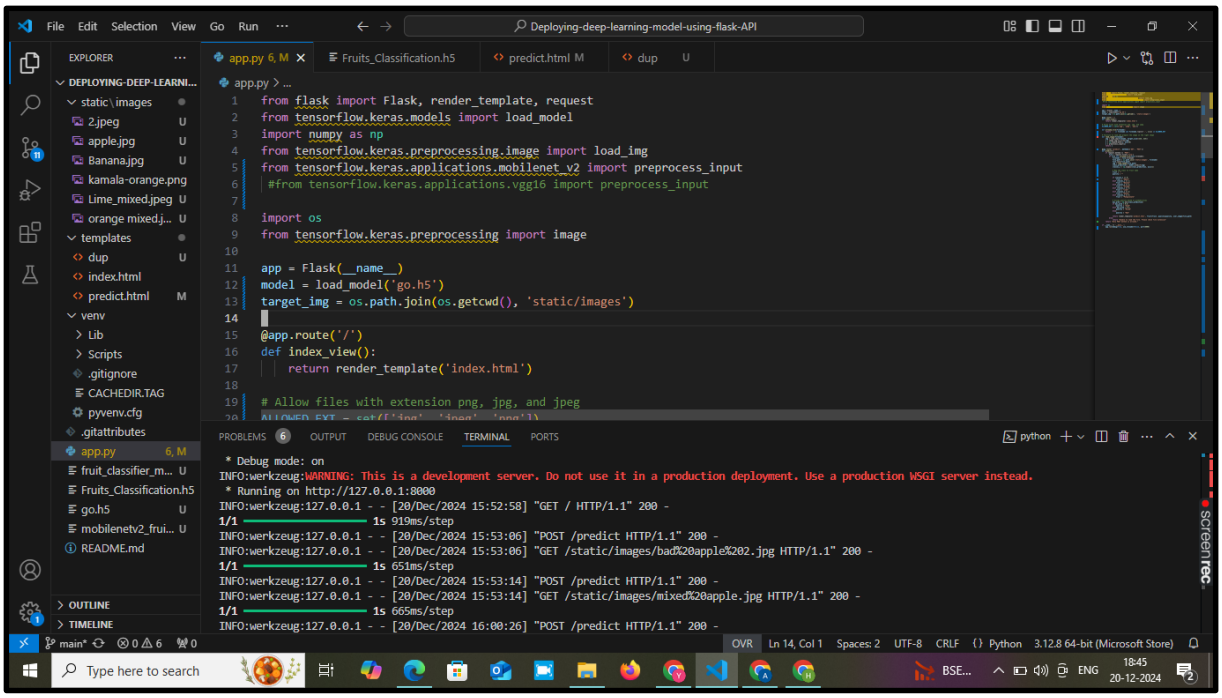

Fig 6.2.1 Execution Procedure

7.TESTING

Testing is the **process of evaluating a system or its components** to ensure that it functions correctly, meets specified requirements, and is free from defects. It involves executing code, validating outputs, and identifying potential errors to improve software quality and reliability.

7.1 Types of Testing

1) **Unit Testing**
2) **Integration Testing**
3) **System Testing**

Unit Testing

- Tests individual components or modules in isolation.
- Ensures feature extraction, classification, and preprocessing work correctly in MobileNetV2.

Integration Testing

- Verifies interactions between different system components.
- Ensures smooth data flow between **Flask API, image processing, and model inference**.

System Testing

- Evaluates the entire system for functionality and performance.
- Includes **stress testing, inference time measurement, and accuracy validation** in real-world conditions.

7.2 Testing

7.2.1. Unit Testing

Unit testing involves testing **individual components or modules** of the system in isolation to ensure they function correctly before integration. In the case of MobileNetV2, unit testing is applied to different layers such as **Conv2D, Dense, Dropout, and Activation layers**, as well as the **preprocessing pipeline** to validate feature extraction, classification, and stability.

Unit Testing Implementation

- **Test the Model Initialization**

```
import tensorflow as tf
from tensorflow.keras.applications import MobileNetV2
def test_model_initialization():
    model = MobileNetV2(weights='imagenet', include_top=False)
    assert model is not None, "Model failed to initialize"
test_model_initialization()
```

- **Test Image Preprocessing**

```
import numpy as np
import cv2
def test_image_preprocessing():
    image = np.random.randint(0, 255, (256, 256, 3), dtype=np.uint8)  # Mock
image
    processed_image = cv2.resize(image, (224, 224))  # Expected resizing
    assert processed_image.shape == (224, 224, 3), "Image preprocessing
failed"
test_image_preprocessing()
```

- **Test Classification Output Format**

```
def test_model_output():
    model = MobileNetV2(weights='imagenet', include_top=True)
    dummy_input = np.random.rand(1, 224, 224, 3)  # Random input
    output = model.predict(dummy_input)
    assert output.shape[1] == 1000, "Model output does not match expected
class count"
test_model_output()
```

7.2.2 Integration Testing

Integration testing verifies that different system components work **seamlessly together**. For MobileNetV2, integration testing ensures that **Flask API**, **image**

processing modules, and **model inference** correctly interact, guaranteeing smooth data flow from **image upload to classification output**.

- **Test Flask API Response**

```
def test_flask_api():
    url = "http://127.0.0.1:5000/predict"
    files = {'file': open('test_image.jpg', 'rb')}
    response = requests.post(url, files=files)
    assert response.status_code == 200, "Flask API did not respond correctly"
test_flask_api()
```

- **Test Image Processing via API**

```
from flask import Flask, request
import cv2
import numpy as np
app = Flask(__name__)
@app.route('/predict', methods=['POST'])
def predict():
    file = request.files['file']
    image = cv2.imdecode(np.fromstring(file.read(), np.uint8),
cv2.IMREAD_COLOR)
    assert image is not None, "Image processing failed"
    return {"message": "Success"}
```

- **Test Model Prediction via API**

```
def test_model_prediction():
    url = "http://127.0.0.1:5000/predict"
    files = {'file': open('test_image.jpg', 'rb')}
    response = requests.post(url, files=files)
    result = json.loads(response.text)
    assert "prediction" in result, "API response missing prediction key"
test_model_prediction()
```

7.2.3. System Testing

System testing evaluates the **entire application** to ensure it meets all functional and performance requirements. For MobileNetV2, system testing involves **stress testing API requests, measuring inference time on GPUs, handling edge cases, and verifying model accuracy** to confirm that the complete fruit classification and grading system operates efficiently in real-world scenarios.

- **Stress Test API with Multiple Requests**

```
import threading

def send_request():
    url = "http://127.0.0.1:5000/predict"
    files = {'file': open('test_image.jpg', 'rb')}
    response = requests.post(url, files=files)
    print(response.json())
threads = []
for _ in range(10):  # Simulating 10 concurrent users
    thread = threading.Thread(target=send_request)
    threads.append(thread)
    thread.start()
for thread in threads:
    thread.join()
```

- **Measure Inference Time**

```
import time
def test_inference_time():
    model = MobileNetV2(weights='imagenet', include_top=True)
    dummy_input = np.random.rand(1, 224, 224, 3)
    start_time = time.time()
    model.predict(dummy_input)
    end_time = time.time()
    assert (end_time - start_time) < 0.1, "Inference time too slow"
test_inference_time()
```

8.RESULT ANALYSIS

8.1 EVALUATION ON MOBILENETV2

TABLE 8.1.1 Classification report for the fruit type model on the test set.

CLASS	**PRECISION**	**RECALL**	**F1-SCORE**	**SUPPORT**
Lime_bad	**0.98**	**0.99**	**0.99**	**204**
Guava_bad	**1.00**	**0.99**	**0.99**	**209**
Orange_bad	**0.99**	**0.98**	**0.99**	**244**
Pomegranate_bad	**1.00**	**0.99**	**1.00**	**252**
Banana_bad	**0.99**	**1.00**	**0.99**	**215**
Apple_bad	**1.00**	**0.99**	**0.98**	**237**
Lime_good	**0.99**	**1.00**	**0.99**	**222**
Apple_good	**1.00**	**0.99**	**0.98**	**232**
Banana_good	**0.99**	**1.00**	**0.99**	**220**
Orange_good	**0.98**	**0.99**	**0.97**	**223**
Pomogranate_good	**1.00**	**1.00**	**1.00**	**1241**
Guava_good	**0.99**	**0.99**	**0.99**	**216**
Apple_mixed	**1.00**	**1.00**	**1.00**	**20**
Guava_mixed	**1.00**	**1.00**	**1.00**	**17**
Orange_mixed	**1.00**	**0.97**	**0.98**	**27**
Lemon_mixed	**0.99**	**0.98**	**0.99**	**55**
Banana_mixed	**0.98**	**0.99**	**0.98**	**49**
Pomogranate_mixed	**1.00**	**0.98**	**0.98**	**23**
Accuracy			**0.99**	**3906**
Macro avg	**0.99**	**0.99**	**0.99**	**3906**
Weighted avg	**0.99**	**0.99**	**0.99**	**3906**

TABLE 8.1.2 Classification report for the fruit quality model on the test set

FRUIT	PRECISION	RECALL	F1-SCORE	SUPPORT
Bad_Quality	**0.99**	**0.99**	**0.99**	**552**
Good_Quality	**1.00**	**0.99**	**1.00**	**908**
Mixed_Quality	**0.98**	**0.99**	**0.99**	**112**
Accuracy			**0.99**	**3906**
Macro avg	**0.99**	**0.99**	**0.99**	**3906**
Weighted avg	**0.99**	**0.99**	**0.99**	**3906**

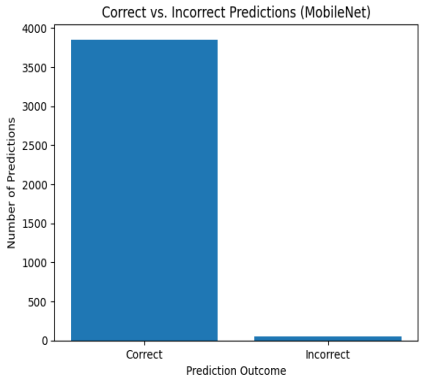

Fig 8.1.1 Predictions of MobileNetV2

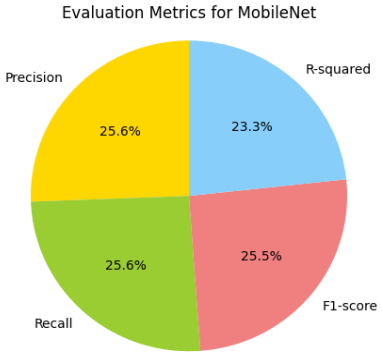

Fig 8.1.2 Evaluation Metrics for MobileNetV2

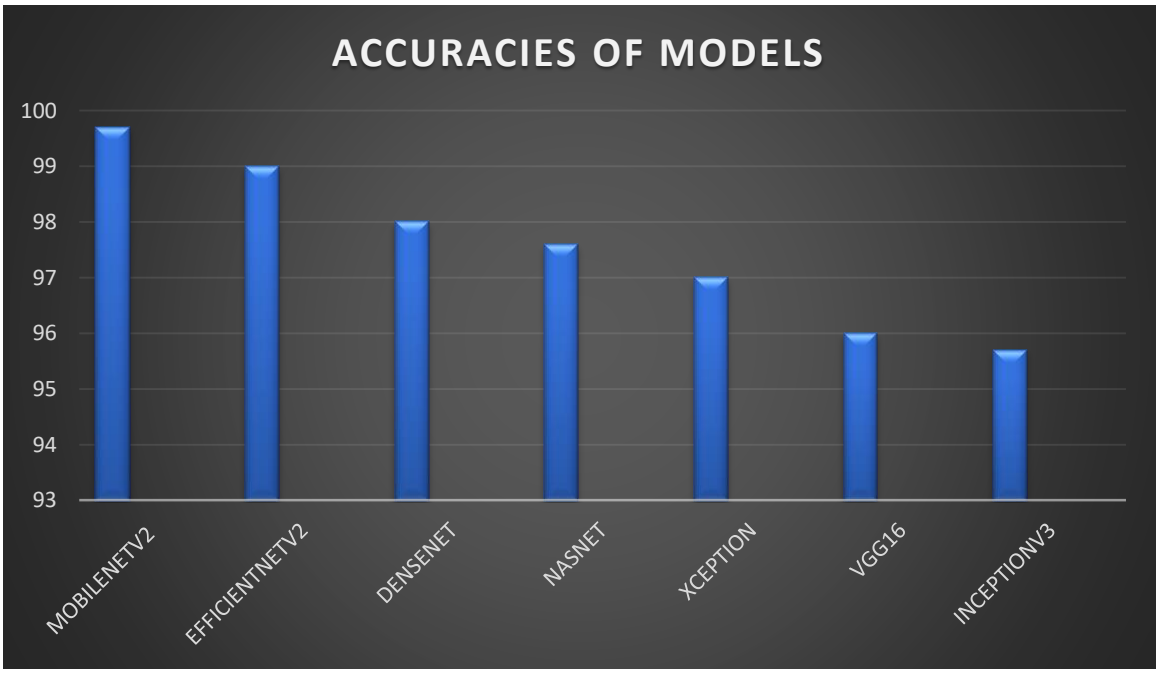

Fig 8.1.3 Accuray of all models

Table 8.1.4 Proposed Approach

Method	**Accuracy**
Mobilenetv2	**99.7%**(PROPOSEDAPPROACH)✓
Efficientnetv2	**99.5%**
Densenet	**98%**
NASNET	**97.6%**
Xception	**97%**
VGG16	**96%**
Inceptionv3	**95.75%**

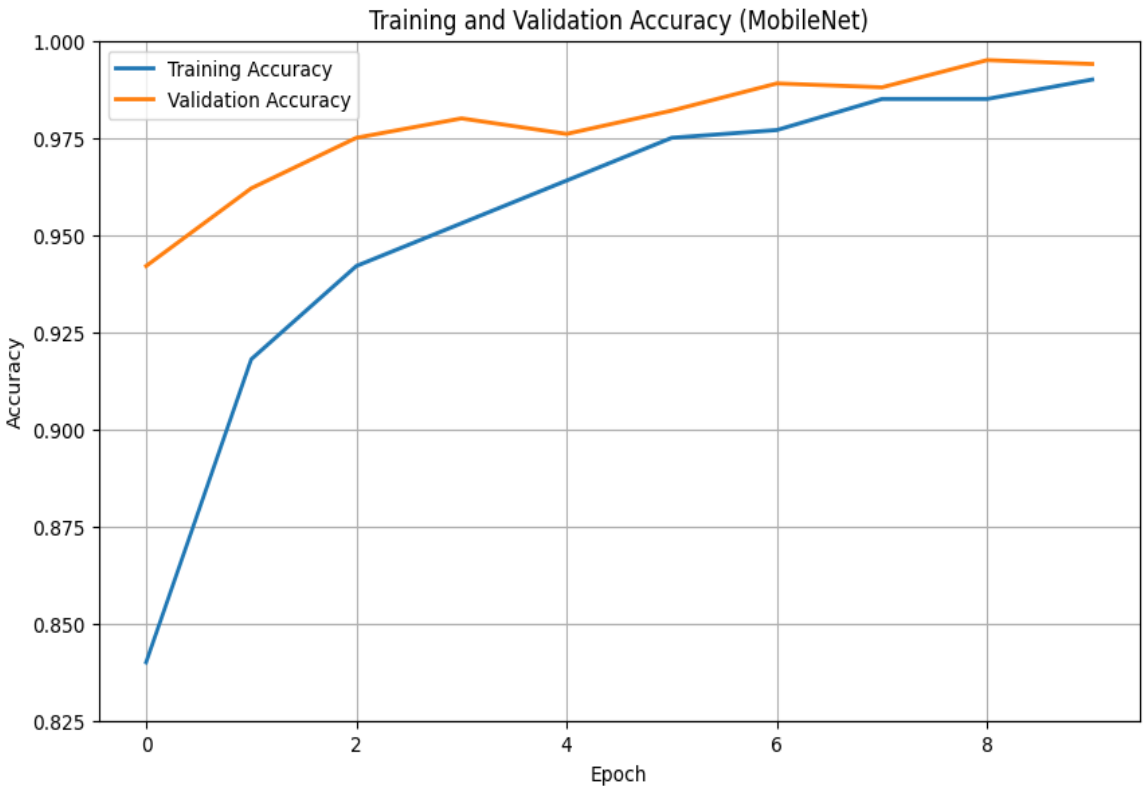

FIG 8.1.5 TRAINING AND VALIDATION ACCURACY

In fig1, it can be observed that the training and validation accuracies of MobileNet rise up to 99% in 10 epochs with closely aligned curves. It is a sign of good generalization without overfitting.

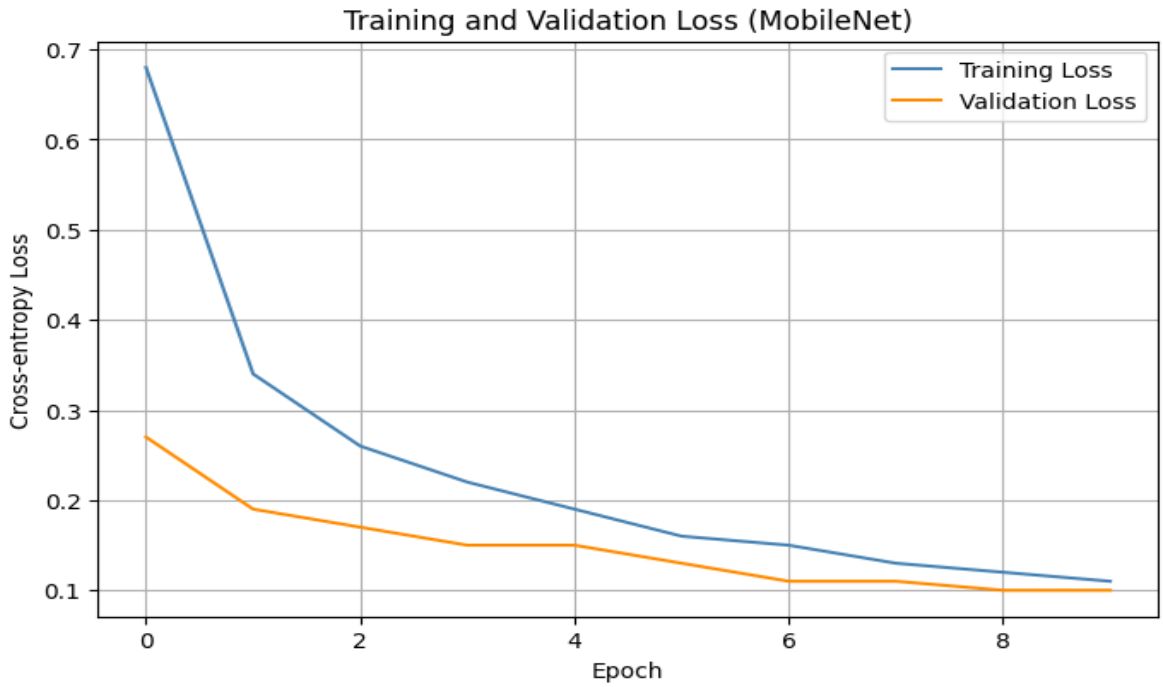

FIG 8.1.6 TRAINING AND VALIDATION LOSS

In fig2, The overall reduction of training and validation losses signifies that the MobileNet model learns well, and both losses are converging to 0.1. Since always the loss for validation is less than that of training, this hints at good generalization and no signs of overfitting.

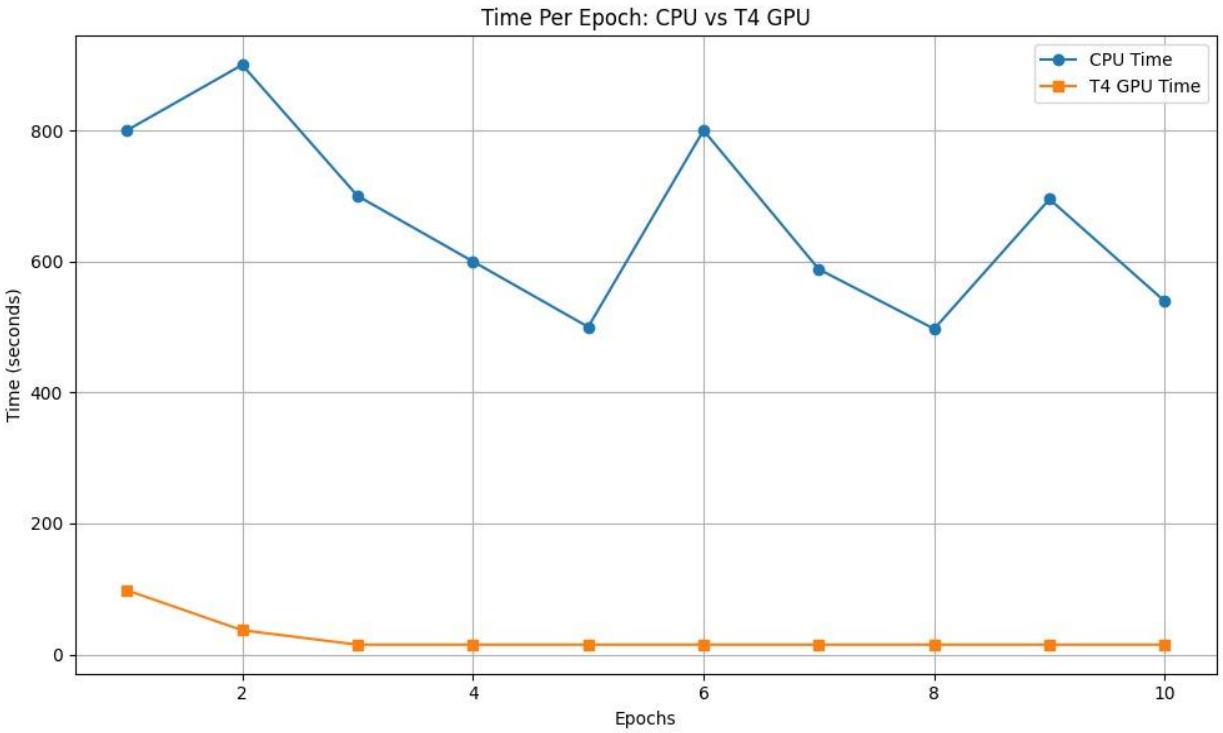

FIG 8.1.7 TIME PER EPOCH : CPU VS T4 GPU

The fig depicts that, T4 GPU significantly reduces the training time per epoch compared to the CPU, speeding up model development and testing. This makes it important for high-performance machine learning experiments.

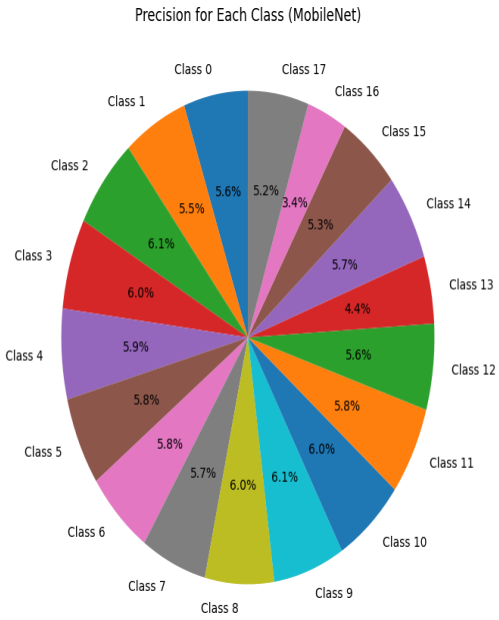

FIG 8.1.8 Bar chart depicts precision for each class

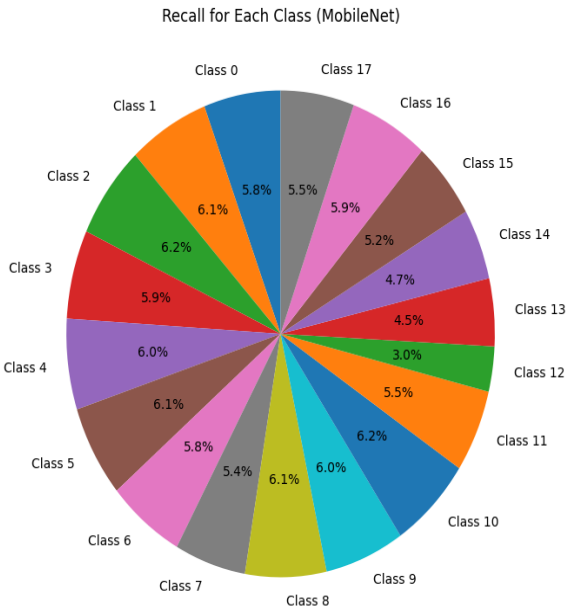

FIG 8.1.9 Bar chart depicts recall for each class

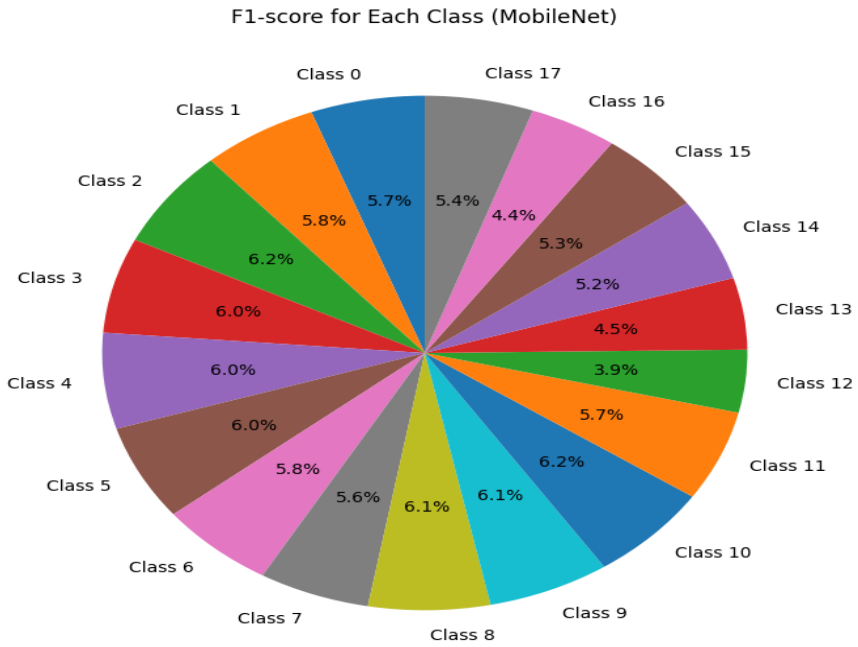

FIG 8.1.10 Bar chart depicts F1-SCORE for each class

The below figures are **pie charts** that represents the **precision , recall and F1-Score for each class** using the MobileNet model. Each segment of the pie chart corresponds to a different class (Class 0 to Class 17), with percentages indicating the precision values for each class. The title confirms that the chart specifically focuses on precision.

9.OUTPUT SCREENS

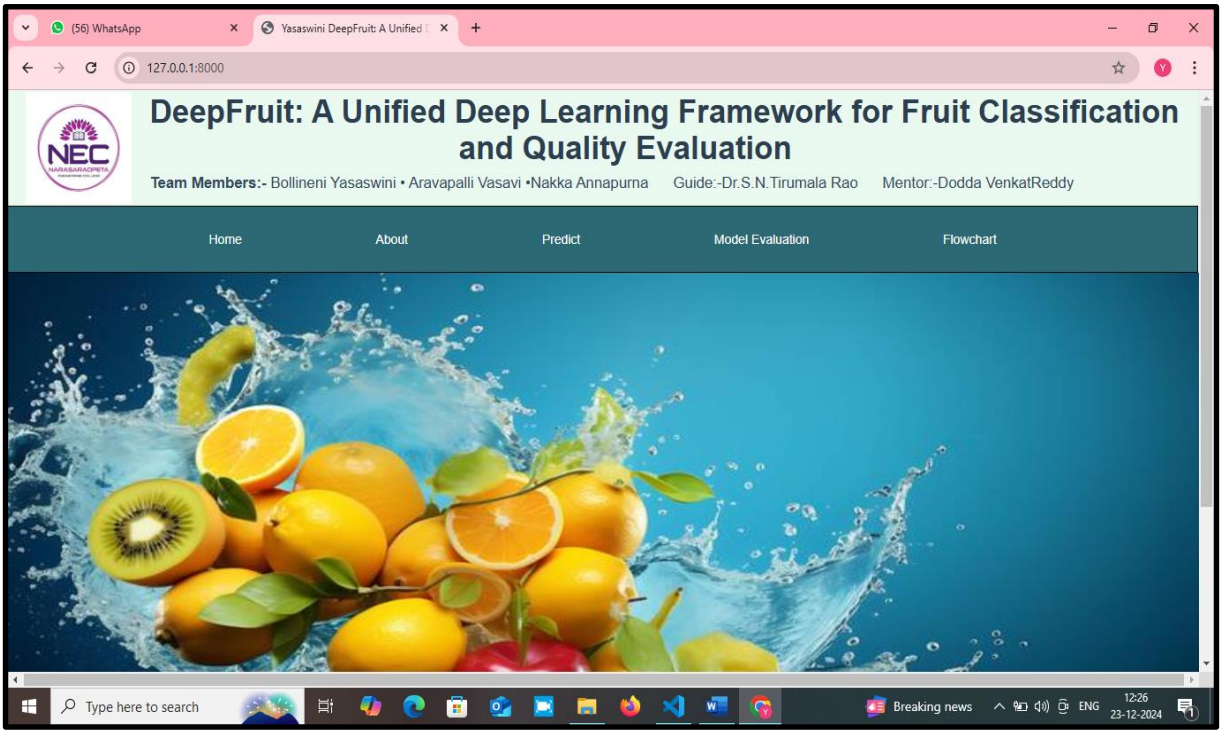

FIG 9.1 HOME PAGE

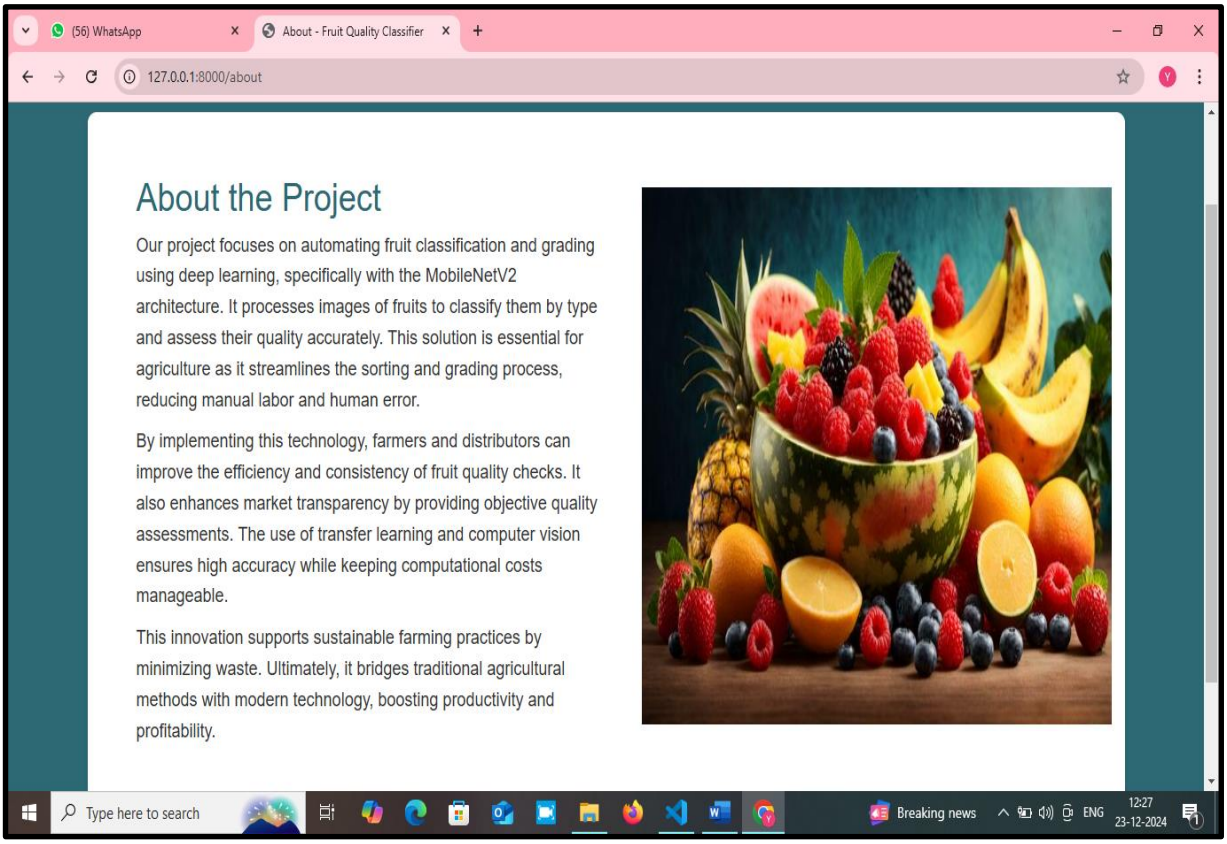

FIG 9.2 ABOUT PAGE

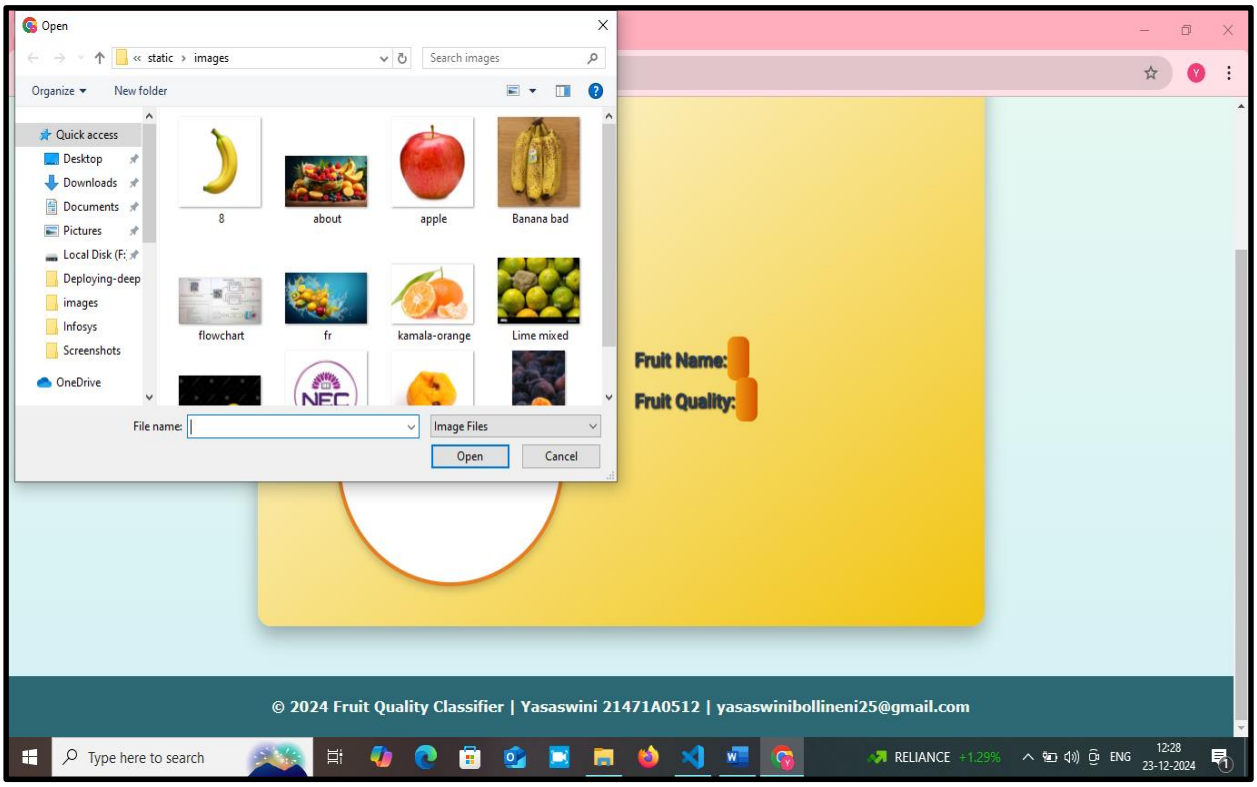

FIG 9.3 PREDICT.HTML

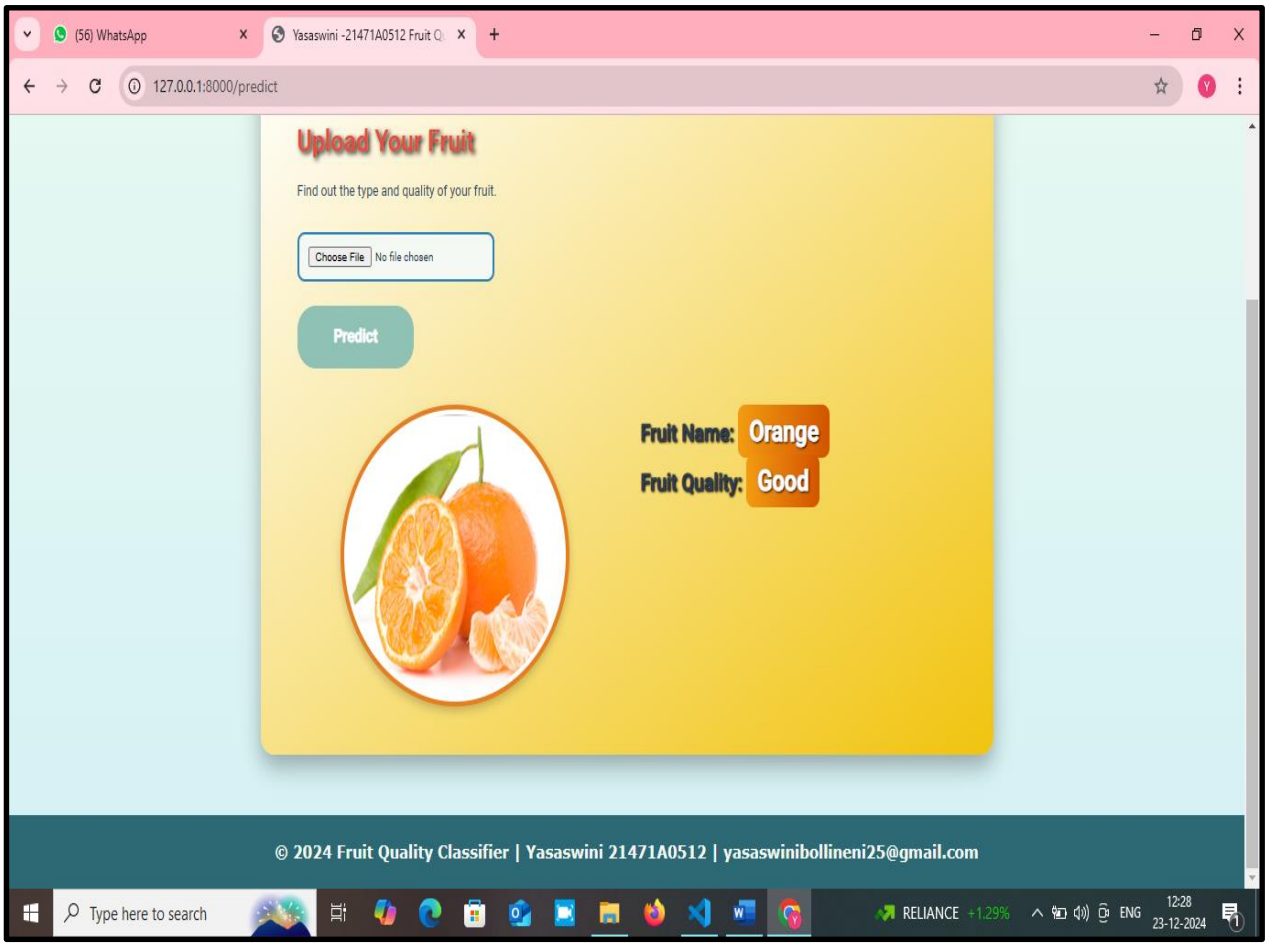

FIG 9.4 OUTPUT PAGE_1

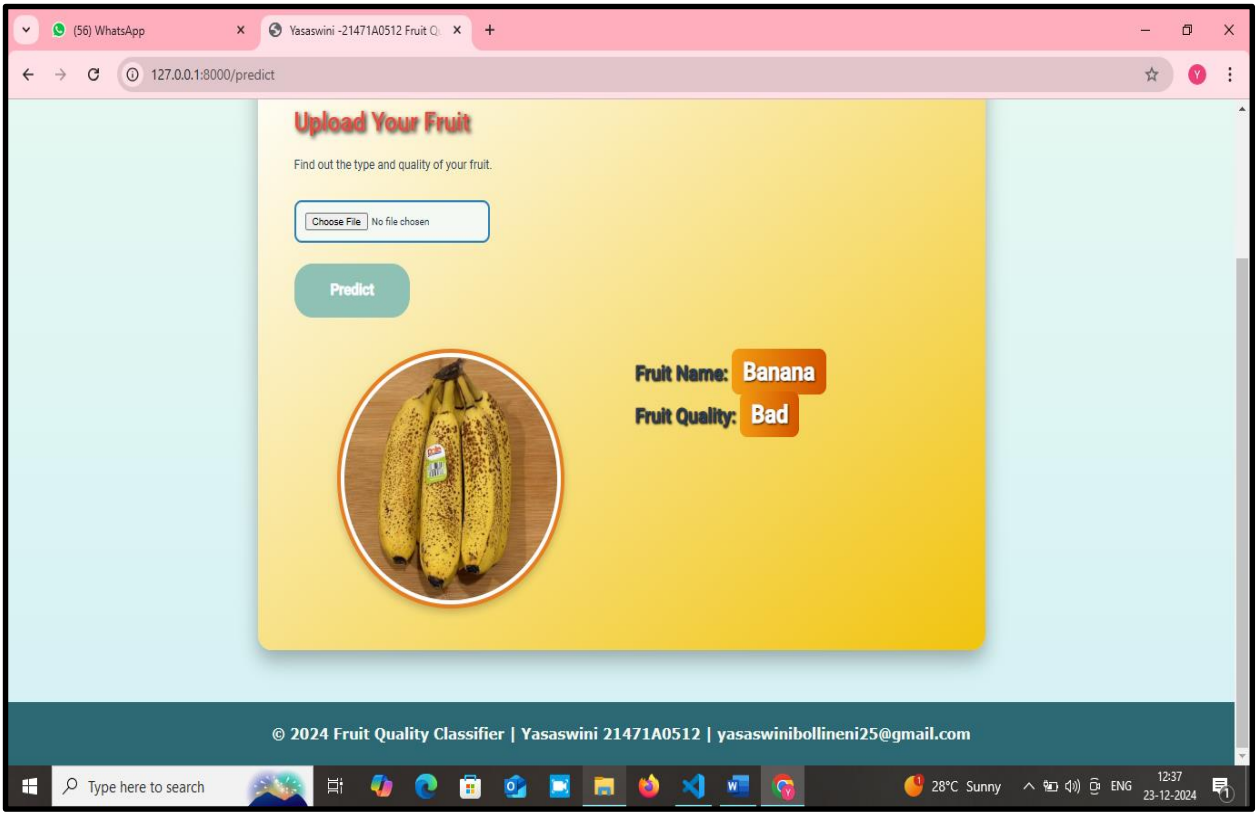

FIG 9.5 OUTPUT PAGE_2

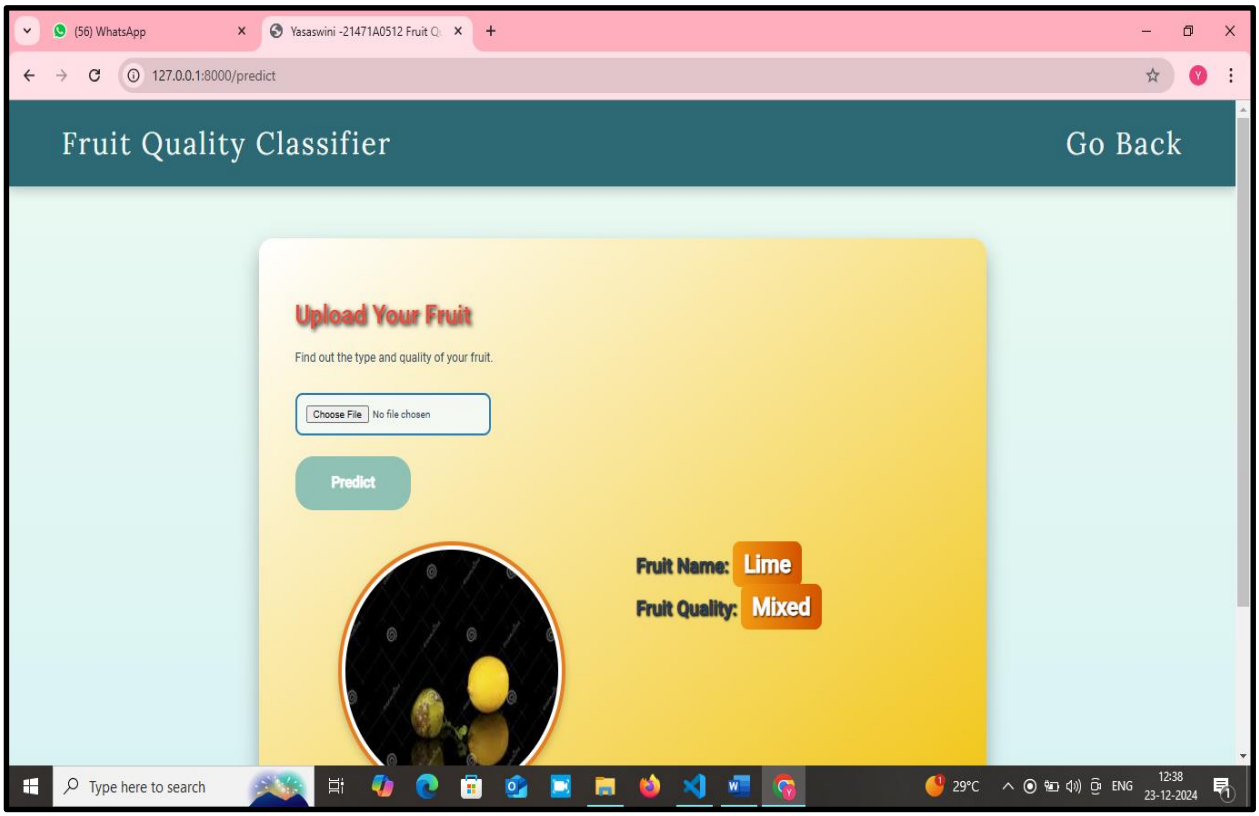

FIG 9.6 OUTPUT PAGE_3

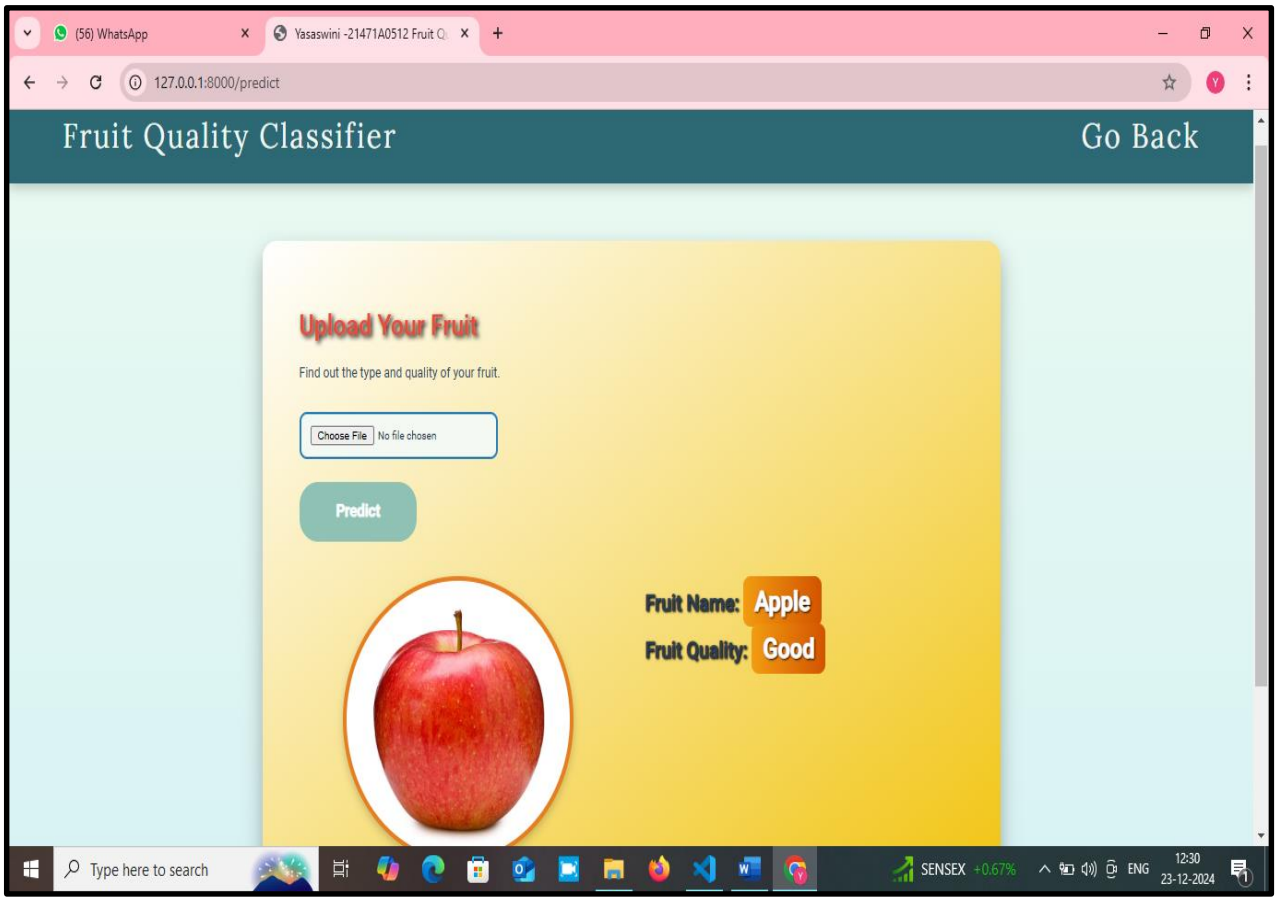

FIG 9.7 OUTPUT PAGE_4

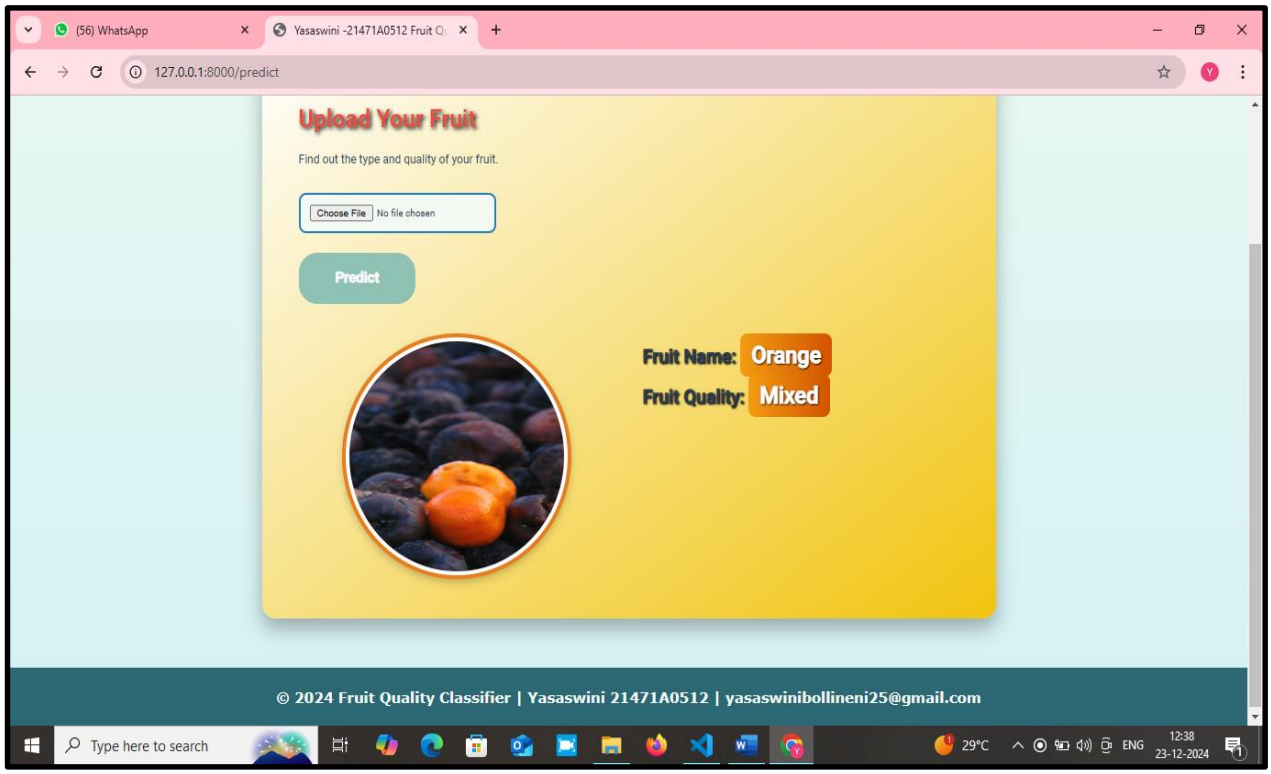

FIG 9.8 OUTPUT PAGE_5

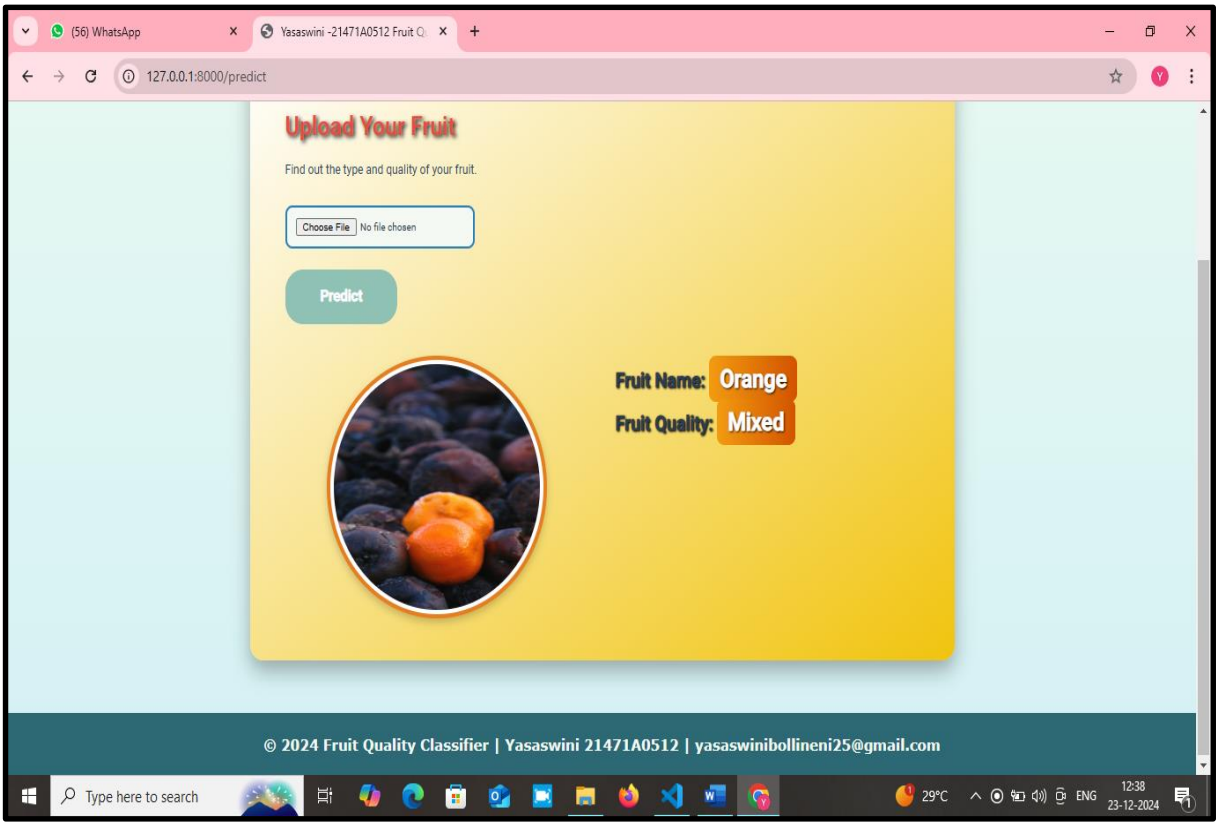

FIG 9.9 OUTPUT PAGE_6

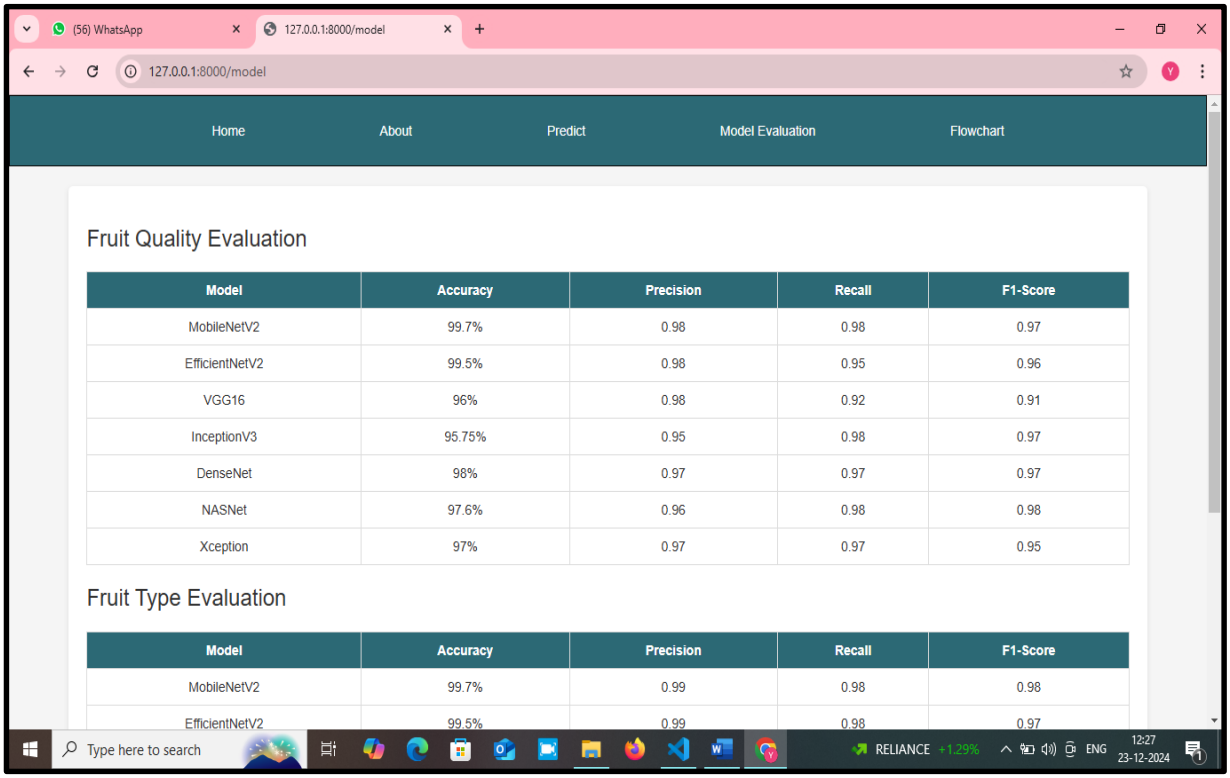

FIG 9.10 EVALUATION PAGE

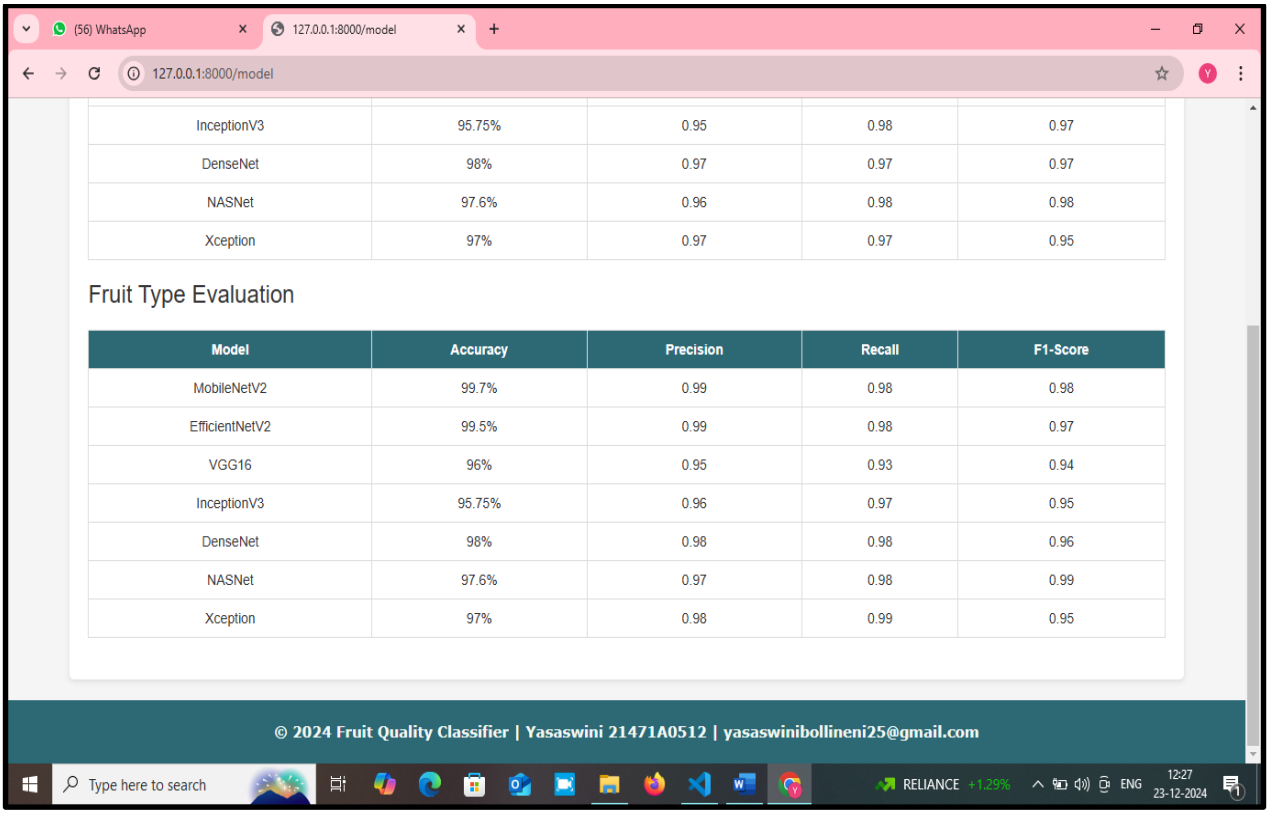

FIG 9.11 EVALUATION PAGE_2

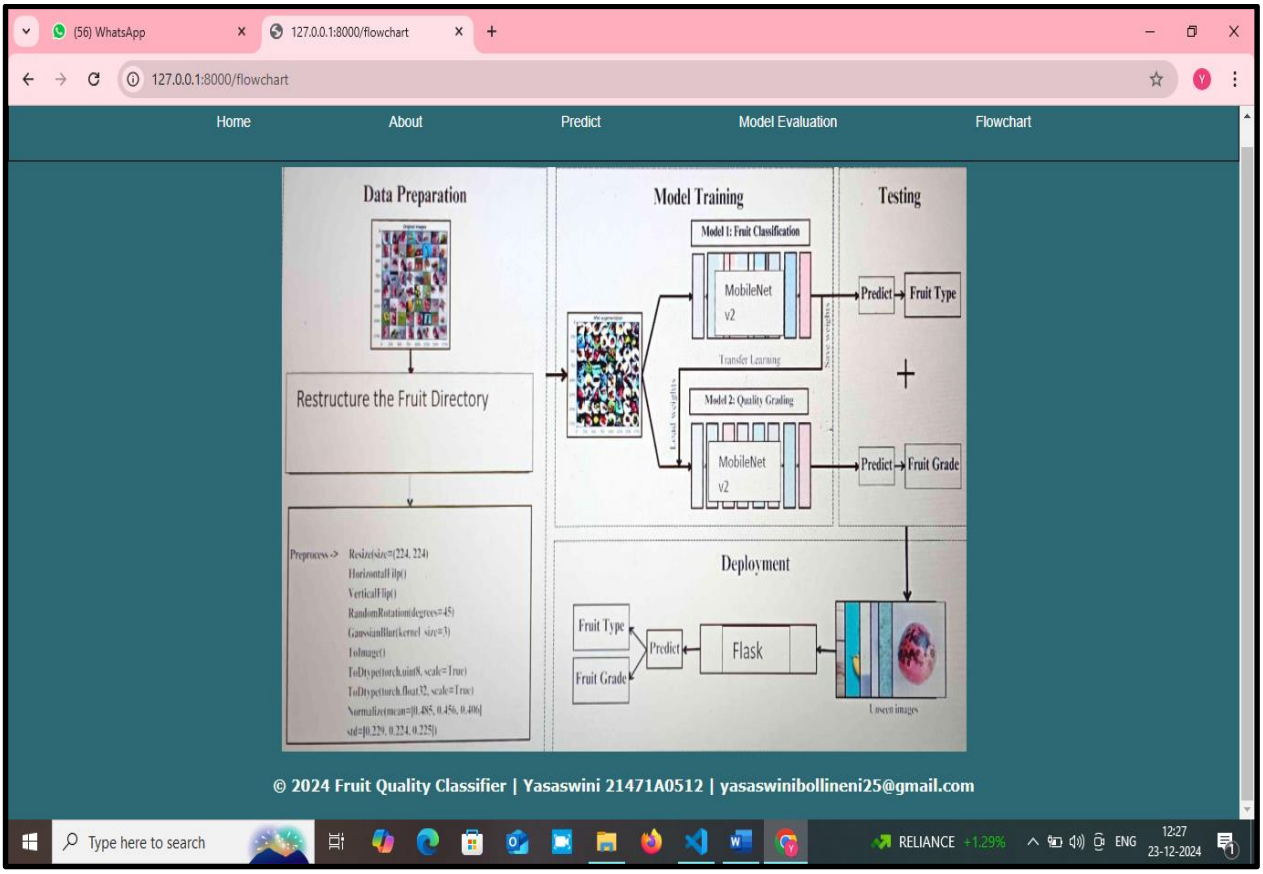

FIG 9.12 FLOWCHART PAGE

10. CONCLUSION

To sum it up, MobileNetV2 occupies the highest position with regard to fruit classification and quality assessment as supported by agricultural mechanization according to our research. Using this efficient design whose large number of tunable weights were utilized optimally, out of tested machine models in experiments conducted, it attained accuracy levels of up to 99.7% which is far above other cutting-edge models such as EfficientNetV2, DenseNet and NASNet but had equivalent or even more tunable weights but failed to reach those accuracies. The proposed twin-model framework, powered by deep learning and coupled with efficient data preprocessing, successfully classified a number of varieties of fruits and their quality. The results presented herein clearly explain that MobileNetV2's architecture with transfer learning and judicious data preprocessing makes for a very strong solution toward betterment in machine vision systems related to agriculture. The continued automation of agricultural practices makes the real-time classification and grading of fruits quite important. The success of MobileNetV2 in this study proves it to be quite ideal for integration into such systems; therefore, opening up possibilities for more efficient and more expandable agricultural processes. This work reflects the promise of advanced deep-learning models in bridging gaps in cultivation and agricultural automation.

11. FUTURE RESEARCH

• Evaluation of the embedded model in an automated machine vision system will help achieve fruit recognition and grading in real-time implementation.

• The improvement of fruit segmentation techniques will increase the capability of the model, focusing only on the important features of the image, hence reducing the influence of irrelevant features from the background.

•Model development technologies were used to analyze images with multiple images of different quality and classify them as either “Good Quality” or “Bad Quality.” In another approach, quality may also be expressed as a percentage that indicates the overall condition of a batch of fruits.

12. REFERENCES

[01] LAMA A. ALDAKHIL and AESHAH A. ALMUTAIRI "Multi-Fruit Classification and Grading Using a Same-Domain Transfer Learning Approach"doi 10.1109/ACCESS.2024.3379276
.

[02] FARSANA SALIM , FAISAL SAEED , SHADI BASURRA , SULTAN NOMAN QASEM AND TAWFIK AL-HADHRAMI "DenseNet-201 and Xception Pre-Trained Deep Learning Models for Fruit Recognition", https://doi.org/10.3390/ electronics12143132, 19 July 2023.

[03] BENJAMIN OLUWAMUYIWA OLORUNFEMI , NNAMDI I. NWULU , OLUWAFEMI AYODEJI ADEBO ,KOSMAS A. KAVADIAS, " Advancements in machine visions for fruit sorting and grading: A bibliometric analysis, systematic review, and future research directions" , DOI :- https://doi.org/10.1016/j.jafr.2024.101154 , April 2024.

[04] WEIWEI ZHANG, "Automated Fruit Grading in Precise Agriculture using You Only Look Once Algorithm", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 14, No. 10, 2023.

[05] SURYA KANT PAL, SUYOG KHANAL, PREM SHANKAR JHA AND RITA ROY, "A Non-Destructive Deep Learning Technique for Fresh and Rotten Fruits Classification", Vol.14 / Issue 78 / June / 2023, ISSN: 0976 – 0997.

[06] KAITAI GUO, HONGLIANG CHEN, YANG ZHENG, QIXIN LIU, SHENGHAN REN , HAIHONG HU AND JIMIN LIANG ,"Efficient Adaptive Incremental Learning for Fruit andVegetable Classification" , DOI:- https://doi.org/10.3390/agronomy14061275. JUNE 2024.

[07] KATHIRESAN SHANKAR, SACHIN KUMAR, ASHIT KUMAR DUTTA, AHMED ALKHAYYAT ,ANWAR JA'AFAR MOHAMAD JAWAD , ALI HASHIM ABBAS AND YOUSIF K. YOUSIF ," An Automated Hyperparameter Tuning Recurrent Neural Network Model for Fruit Classification" , DOI:- https://doi.org/10.3390/math10132358, JULY 2022.

[08] JIA-JI WANG," Recognition system for fruit classification based on 8-layer convolutional neural network", DOI:- 10.4108/eai.17-2-2022.173455 , FEB 2022.

[09] KHADIJA MUNIR, ARIF IQBAL UMAR, AND WAQAS YOUSAF, "Automatic Fruits Classification System Based on Deep Neural Network", DOI :- https://doi.org/10.24949/njes.v13i1.501 Vol.13,No.1,2020.

[10] OMERFAROOQ,JASMEEN GILL"Vegetable Grading and Sorting using Artificial Intelligence" DOI Link: https://doi.org/10.22214/ijraset.2022.40407

[11] APOLO-APOLO, O. E., MARTÍNEZ-GUANTER, J., EGEA, G., RAJA, P., & PÉREZ-RUIZ, M. "Deep learning techniques for estimation of the yield and size of citrus fruits using a UAV. ". DOI:- https://doi.org/10.1016/j.eja.2020.126030.

[12] GUICHAO LIN ,YUNCHAO TANG,XIANGJUN ZOU ,CHENGLIN WANG "Three-dimensional reconstruction of guava fruits and branches using instance segmentation and geometry analysis"
DOI :- https://doi.org/10.1016/j.compag.2021.106107.

[13] MOHAMMAD MOMENY,AHMAD ,YU-DONG ZHANG JAHANBAKHSHI,KHALEGH JAFARNEZHAD "Accurate classification of cherry fruit using deep CNN based on hybrid pooling approach". DOI :- https://doi.org/10.1016/j.postharvbio.2020.111204

[14] MURE ̧SAN, H.; OLTEAN, "M. Fruit recognition from images using deep learning". arXiv 2021, arXiv:1712.00580. Available online: https://arxiv.org/abs/1712.00580 (accessed on 1 June 2023). Volume 10, Number 1, 2018

[15] ANUJA BHARGAVA, ATUL BANSAL & VISHAL GOYAL "Machine Learning–Based Detection and Sorting of Multiple vegetables and Fruits" 30 August 2021,Volume 15 DOI :https://link.springer.com/article/10.1007/s12161-021-02086-1

[16] HAFSA RAISSOULI , ABRAR ALI ALJABRI, SARAH MOHAMMED ALJUDAIBI , FAZILAH HARON , GHADA ALHARBI "Date Grading using Machine Learning Techniques on a Novel Dataset"
DOI :- 10.14569/IJACSA.2020.0110893

[17] DHIYA MAHDI ASRINY, SEPTIA RANI AND AHMAD FATHAN HIDAYATULLAH "Orange Fruit Images Classification using Convolutional Neural Networks" doi :- 10.1088/1757-899X/803/1/012020.

[18] Y. ZHANG, L. WU, Classification of fruits using computer vision and a multiclass support vector machine. Sensors **12**(9), 12489–12505 (2012)

[19] H. CHENG, L. DAMEROW, Y. SUN, M. BLANKE, Early yield prediction using image analysis of apple fruit and tree canopy features with neural networks. J. Imaging **3**, 6 (2017)

[20] H. MUREŞAN, M. OLTEAN, Fruit recognition from images using deep learning. Acta Universitatis Sapientiae. Informatica **10**(1), 26-42

[21] M. S. HOSSAIN, M. AL-HAMMADI AND G. MUHAMMAD, "Automatic Fruit Classification Using Deep Learning for Industrial Applications," in IEEE

Transactions on Industrial Informatics, vol. 15, no. 2, pp. 1027-1034, Feb. 2019, doi: 10.1109/TII.2018.2875149.

[22] A. PANDE, M. MUNOT, R. SREEEMATHY AND R. V. BAKARE, "An Efficient Approach to Fruit Classification and Grading using Deep Convolutional Neural Network," 2019 IEEE 5th International Conference for Convergence in Technology (I2CT), Bombay, India, 2019, pp. 1-7, doi: 10.1109/I2CT45611.2019.9033957.

[23] H. B. ÜNAL, E. VURAL, B. K. SAVAŞ AND Y. BECERIKLI, "Fruit Recognition and Classification with Deep Learning Support on Embedded System (fruitnet)," 2020 Innovations in Intelligent Systems and Applications Conference (ASYU), Istanbul, Turkey, 2020, pp. 1-5, doi: 10.1109/ASYU50717.2020.9259881.

[24] I. M. NASIR, A. BIBI, J. H. SHAH, M. A. KHAN, M. SHARIF, K. IQBAL, Y. NAM, AND S. KADRY, ‘‘Deep learning-based classification of fruit diseases: An application for precision agriculture,’’ Comput., Mater. Continua, vol. 66, no. 2, pp. 1949–1962, 2021.

[25] M. KNOTT, F. PEREZ-CRUZ, AND T. DEFRAEYE, ‘‘Facilitated machine learning for image-based fruit quality assessment,’’ J. Food Eng., vol. 345, May 2023, Art. no. 111401.

[26] M. SANDLER, A. HOWARD, M. ZHU, A. ZHMOGINOV, AND L.-C. CHEN, ‘‘MobileNetV2: Inverted residuals and linear bottlenecks,’’ in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Salt Lake City, UT, USA, Jun. 2018, pp. 4510–4520.