DEEP FRUIT VEG: AUTOMATED FRUIT AND VEG IDENTIFICATION

AN INDUSTRY ORIENTED MAJOR REPORT

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

POTHURAJU DAVID **21UK1A05M3**

Under the guidance of

Mr. G. Satish Chander

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINERING COLLEGE

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL(T.S)-506005

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE(WARANGAL)

CERTIFICATE OF COMPLETION
INDUSTRY ORIENTED MAJOR PROJECT

This is to certify that the UG Project Phase-1 entitled "**DeepFruitVeg: Automated Fruit And Veg Identification"** is being submitted by KUDTHALA SRIKANTH (21UK1A05Q1),POTHURAJU DAVID(21UK1A05M3), KARRE NAGARAJU (21UK1A05M8),MAMIDI ABHIRAM (21UK1A05Q6) in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering to Jawaharlal Nehru Technological University Hyderabad during the academic year 2023-2024

Project Guide	**HOD**
Mr. G. Satish Chander	**Dr. R. Naveen Kumar**
(Assistant Professor)	(Professor)

EXTERNAL

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr . SYED MUSTHAK AHAMED**, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG Project Phase- 1 in the institute.

We extend our heartfelt thanks to **Dr. R. NAVEEN KUMAR**, Head of the Department of CSE, Vaagdevi Engineering College for providing us necessary infrastructure and there by giving us freedom to carry out UG Project Phase-1.

We express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the UG Project Phase-1.

We express heartfelt thanks to the guide, **Mr. G. Satish Chander**, Assistant professor, Department of CSE for his constant support and giving necessary guidance for completion of this UG Project Phase1.

Finally, we express our sincere thanks and gratitude to my family members, friends for their encouragement and outpouring their knowledge and experience throughout the thesis.

POTHURAJU DAVID **21UK1A05M3**

ABSTRACT

- This analysis explores customer reviews of cell phones on Amazon, focusing on key patterns, sentiments, and factors influencing customer satisfaction. By using natural language processing (NLP) techniques, including sentiment analysis and topic modeling, we categorize reviews to understand positive and negative themes. Our study reveals that while factors like battery life, camera quality, and performance are most commonly discussed, sentiment around price and durability significantly affects overall ratings. This analysis aims to provide insights for manufacturers to improve product offerings and for potential customers to make informed purchasing decisions based on aggregated user feedback. Through this investigation, we demonstrate how consumer opinions shape brand perceptions and influence purchasing behaviour on e-commerce platforms.

- This analysis delves into a dataset of Amazon cell phone reviews, examining customer feedback through a comprehensive lens to understand the determinants of consumer satisfaction and dissatisfaction. Utilizing advanced natural language processing (NLP) techniques such as sentiment analysis, topic modelling, and feature extraction, we categorize and quantify review content, revealing key drivers of positive and negative sentiment. The study identifies recurring themes, including battery life, camera quality, screen display, software performance, and price-value perception, all of which strongly impact user satisfaction. A closer examination of negative reviews highlights issues with product durability, software updates, and customer service, each of which correlates to lower ratings and returns.

- To enhance the granularity of analysis, we segment data by device models, price categories, and brand reputation, uncovering unique insights into customer preferences across demographics. Additionally, comparative sentiment analysis between leading smartphone brands reveals distinct brand-related strengths and weaknesses in customer perceptions. The findings not only provide actionable insights for manufacturers and marketers to refine product development and customer service strategies but also serve as a guide for potential buyers to make more informed purchase decisions. Ultimately, this study demonstrates how customer feedback, when systematically analyzed, can significantly shape product innovation and improve customer satisfaction within the highly competitive e-commerce marketplace.

TABLE OF CONTENTS:-

1. INTRODUCTION

1.1. OVERVIEW:

Deepfruitveg leverages the power of deep learning to transform how fruits and vegetables are identified, categorized, and managed in various industries. This system utilizes advanced image recognition techniques to streamline processes in food processing, retail, and agriculture, ensuring efficiency and accuracy. Through rigorous training on a diverse dataset of fruit and vegetable images, the project achieves high precision in distinguishing between 30 different classes, addressing challenges in real-time classification and scalability.

The methodology incorporates key techniques such as data preprocessing, feature extraction, and multi-class classification, enabling the identification of critical attributes like shape, colour, and texture. The system is fine-tuned to highlight areas of application such as automated sorting in food plants, quality assurance in supermarkets, and crop monitoring in agricultural settings.

Deepfruitveg also examines variations in identification performance across different environmental conditions, such as lighting and background noise, to ensure robustness in real-world applications. Insights drawn from this study provide manufacturers, farmers, and retailers with actionable feedback on improving processes and aligning product offerings with market demands. Ultimately, this project sets the stage for innovation in the food and agriculture industries, driving efficiency while meeting consumer expectations for quality and reliability.

1.2. PURPOSE:

The purpose of Deepfruitveg is to develop an intelligent and automated system for accurately identifying fruits and vegetables using deep learning. This project aims to address challenges in the food and agriculture industries by enhancing efficiency in processes like sorting, quality control, and crop monitoring. By providing a reliable and scalable solution, Deepfruitveg seeks to streamline operations in food processing plants, ensure consistent quality in supermarkets, and empower precision agriculture practices. Ultimately, it bridges the gap between technological advancements and real-world applications, catering to the needs of manufacturers, farmers, and consumers.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM:

The identification and classification of fruits and vegetables pose significant challenges in industries like food processing, retail, and agriculture. Traditional methods, such as manual sorting and visual inspection, are labour-intensive, time-consuming, and prone to human error. These inefficiencies often lead to inconsistent quality control, increased operational costs, and delays in supply chain management.

Moreover, variations in environmental factors, such as lighting, background, and produce condition (e.g., ripeness, damage), can impact the accuracy of existing automated systems. Many current technologies lack scalability, struggle with recognizing diverse fruit and vegetable classes, and fail to adapt to real-world conditions. This gap creates a pressing need for robust, reliable, and scalable solutions to streamline these processes and meet industry demands.

2.2 PROPOSED SOLLUTION:

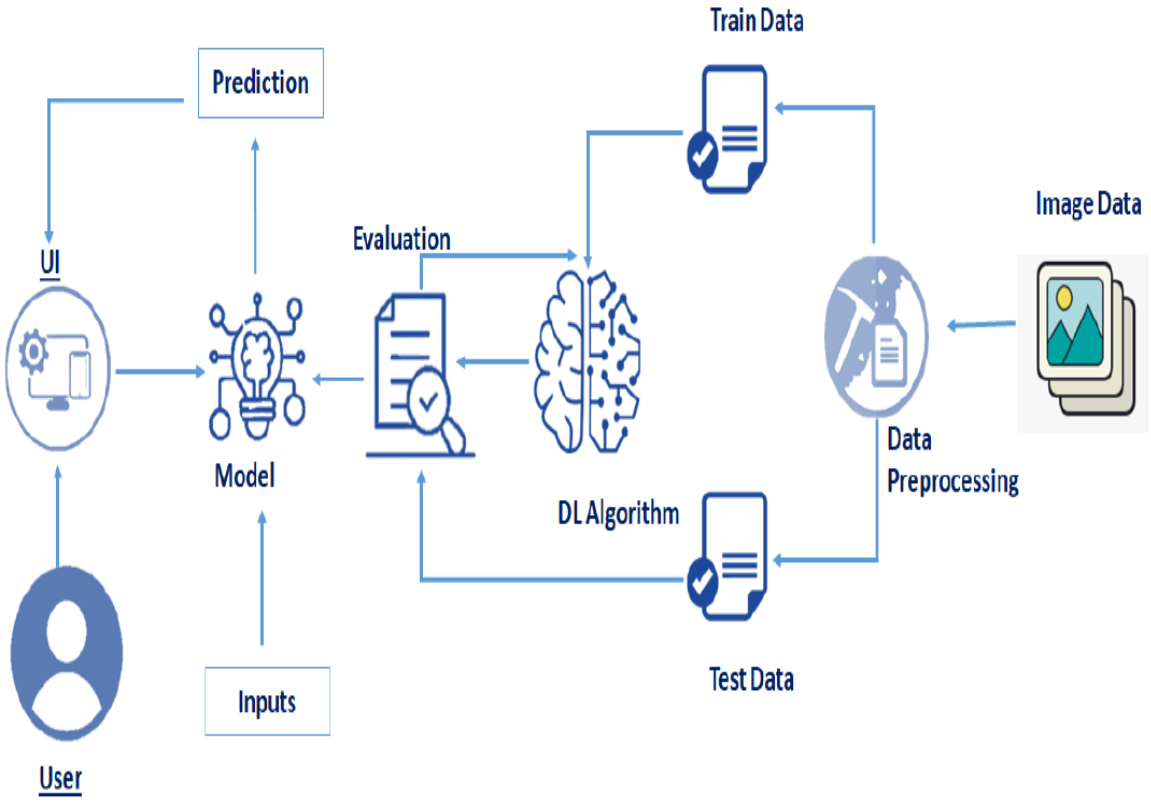
To address the challenges of automating the identification of fruits and vegetables in real-world environments, we propose a robust, multi-step solution utilizing deep learning and computer vision technologies. This solution aims to streamline the process of fruit and vegetable sorting, quality control, and precision agriculture by offering accurate, efficient, and scalable identification systems.

1. **Image Classification using Deep Learning**
   We will train a deep learning model, specifically a Convolutional Neural Network (CNN), to classify fruits and vegetables from images. The model will be trained on a large, annotated dataset of fruit and vegetable images to ensure high accuracy in identifying various species. This layer of analysis allows the system to recognize different produce items in diverse conditions and from multiple angles.
2. **Data Augmentation and Model Optimization**
   To improve the model's performance, we will apply data augmentation techniques such as rotation, flipping, and colour variation to simulate real-world variations in the appearance of fruits and vegetables. These augmentations will increase the model's robustness and reduce overfitting. We will also explore hyperparameter tuning and optimization techniques to enhance model accuracy and reduce inference time.
3. **Real-time Identification and Classification**
   The model will be integrated into a real-time system that can process images taken by cameras in food processing plants, supermarkets, or agricultural fields. This system will automatically detect, classify, and label fruits and vegetables in real time, allowing for fast sorting, inventory management, and quality control. The real-time capabilities ensure operational efficiency in high-volume environments.
4. **Feature Extraction and Quality Assessment**
   Beyond simple classification, the model will also extract key features such as size, color, shape, and ripeness to assess the quality of the produce. This can help identify fruits and vegetables that are under-ripe, overripe, or damaged, allowing for better-quality control and reducing waste in supply chains. We will use regression techniques to predict ripeness levels and other quality indicators based on visual features.
5. **Integration with Automated Sorting Systems**
   To further streamline the process, the identified and categorized produce will be linked to automated sorting systems. This integration allows for the automatic distribution of fruits and vegetables into different quality categories or storage areas based on their classification. The automated sorting system will enhance the speed and accuracy of the sorting process, minimizing human intervention and errors.

6. **Real-time Feedback and Data Analysis for Farmers and Manufacturers**
   We will implement a feedback loop for farmers and manufacturers to monitor the quality and quantity of their produce. The system will generate real-time analytics on yield, quality distribution, and common issues (e.g., pest damage, poor ripeness), helping stakeholders make data-driven decisions for better crop management, harvesting, and packaging.
7. **Scalable Model Deployment and Continuous Learning**
   As new varieties of fruits and vegetables are introduced or as environmental conditions change, the model will be continuously updated with new data. A scalable deployment system will allow for periodic retraining with fresh data to improve accuracy over time. By implementing continuous learning, the system can adapt to new challenges and maintain high performance.
8. **User-Friendly Interface and Visualizations**
   To make the system accessible to both consumers and manufacturers, we will create an intuitive user interface that displays real-time results, sorting outcomes, and quality assessments. Visual dashboards will provide actionable insights, helping farmers and manufacturers optimize their operations and ensuring consumers receive high-quality products.

3.THEORITICAL ANALYSIS

3.1. BLOCK DIAGRAM:



3.2. SOFTWARE DESIGNING:

To effectively implement the Deep fruit veg system, which leverages deep learning for automated fruit and vegetable identification, we propose a software architecture that utilizes modular, scalable components for data ingestion, preprocessing, model training, prediction, and visualization. The system will handle large datasets of fruit and vegetable images, process them in real-time, and generate insights for applications in food processing, quality control, and precision agriculture.

1. Data Ingestion Layer

- **Data Sources**:
  The system will focus on images of fruits and vegetables collected from various sources (e.g., public image datasets, camera systems in agricultural fields, and food processing plants). Data will be ingested through APIs or direct uploads to a centralized data storage system.
- **Data Preprocessing**:
  Image preprocessing will involve resizing, normalization, and augmentation (e.g., random rotations, flipping, colour adjustments) to improve the robustness and accuracy of the model. Metadata such as product name, category, and origin will also be extracted, where applicable. The pre processed data will be organized for efficient use in training and validation.

2. Deep Learning Model Training Layer

- **Model Architecture**:
  The core of the system will be a deep learning model, likely a Convolutional Neural Network (CNN), trained to classify images of fruits and vegetables. This model will be built using popular frameworks like TensorFlow or PyTorch.
- The model will be trained on labeled datasets consisting of images categorized by fruit or vegetable type.
- Transfer learning may be used to leverage pre-trained models (e.g., VGG16, ResNet) to reduce training time and improve model accuracy.
- **Training Pipeline**:
  A robust training pipeline will handle dataset splitting (training, validation, test sets), hyperparameter tuning, and model evaluation. It will include tracking of training progress, loss functions, and accuracy metrics, ensuring continuous improvements in model performance.

3. Prediction and Inference Layer

- **Real-time Prediction**:
  The trained model will be deployed in an inference engine capable of making predictions on new, incoming image data in real time. This can be applied in various use cases, such as identifying fruits and vegetables in food processing plants or quality control systems.
- **Batch Prediction**:
  The system will also support batch processing for scenarios where large volumes of images need to be classified, such as automated sorting of produce in warehouses or farms.

4. Data Storage and Management

- **Database Design**:
  All data (images, predictions, metadata) will be stored in a relational or NoSQL database. Image data will be stored in cloud storage (e.g., AWS S3) or a distributed file system, while structured metadata (e.g., fruit/vegetable type, confidence scores) will be stored in relational tables for easy querying and reporting.
- **Data Indexing and Retrieval**:
  To enable efficient querying and retrieval, an indexing system will be implemented. This will allow quick access to images, predictions, and related data for analysis, reporting, or further processing.

5. Analysis and Aggregation Layer

- **Performance Evaluation**:
  This module will continuously monitor the model's performance based on metrics such as accuracy, precision, recall, and F1 score. It will help identify any degradation in model performance over time and suggest potential retraining or fine-tuning.
- **Data Insights and Reporting**:
  This layer will aggregate predictions and provide detailed analysis reports. For example, it could identify the most frequently detected fruits and vegetables, track the accuracy of predictions by type, and flag common misclassifications. These reports will be valuable for optimizing product sorting in food processing systems and improving overall operational efficiency.

6. Visualization and Reporting Module

- **Interactive Dashboard**:
  A dynamic dashboard will provide real-time insights into the performance of the model, including visualizations of classification results, model accuracy over time, and trends in detected fruit and vegetable types. Users can filter data by category (e.g., fruit or vegetable type), prediction confidence, or accuracy.
- **Prediction Results**:
  The system will display predicted labels, confidence scores, and images in an easy-to-navigate interface. Consumers and manufacturers can use the results to make data-driven decisions, such as selecting the best produce or improving sorting processes in food production.

7. User Interface (UI) and Accessibility

- **Consumer Interface**:
  For non-technical users, such as consumers or workers in food processing plants, a simple UI will be developed to upload images, view predictions, and track product quality over time. It will allow users to easily interact with the system without requiring technical expertise.
- **Manufacturer Interface**:
  A more advanced dashboard will be provided to manufacturers or food processing plant operators, offering detailed insights into model accuracy, trends in fruit and vegetable types, and

performance metrics. This interface will also include tools for improving or retraining the model based on feedback or new data.

8. Scalability and Performance Optimization

- **Microservices Architecture**:
  The system will be designed using a microservices architecture, where each component (data ingestion, model training, prediction, and analysis) operates as an independent service. This approach allows each component to scale independently and ensures the system can handle large volumes of data efficiently.
- **Cloud Deployment**:
  The system will be deployed on a cloud platform (e.g., AWS, Google Cloud) for high availability, scalability, and performance optimization. Cloud services will enable elastic scaling to accommodate fluctuating demand for predictions and training processes.
- **Real-time and Batch Processing**:
  For real-time applications, such as monitoring food production in a factory, the system will support real-time image classification. For less time-sensitive tasks, batch processing will be used to handle large volumes of images, such as in agriculture or supply chain management.

4.EXPERIMENTAL INVESTIGATION:

To evaluate the effectiveness of the Deepfruitveg system for fruit and vegetable identification, we propose a detailed experimental investigation. This investigation will assess the system's performance in terms of accuracy, efficiency, scalability, and usability across various deployment scenarios. The experiments will be designed to validate the system's capabilities in real-world applications, such as food processing, quality control, and precision agriculture.

1. Experimental Setup

- **Data Collection**:
  The dataset for the experiment will consist of a large set of labeled images of fruits and vegetables, covering a wide range of types, shapes, colors, and sizes. Data will be sourced from publicly available datasets (e.g., ImageNet, Fruits 360) and supplemented with images from real-world agricultural and food processing settings.
    - **Train/Test Split**:
      The dataset will be split into training (80%) and testing (20%) subsets to evaluate the model's performance. The training data will be used to train the deep learning model, while the testing data will be reserved for evaluating model accuracy and generalization.
- **System Configuration**:
  The system will be deployed on a cloud-based infrastructure (e.g., AWS, Google Cloud) with GPU support for model training. The model will be implemented using deep learning frameworks such as TensorFlow or PyTorch. The system will be configured for both batch processing and real-time inference, depending on the use case.

2. Performance Evaluation Metrics

- **Accuracy**:
  The primary metric for evaluating the model's performance will be classification accuracy, which is the percentage of correctly classified images out of the total number of images in the test set. This metric will provide an overall measure of how well the model can identify fruits and vegetables.
- **Precision, Recall, and F1-Score**:
  To assess the model's performance in more detail, especially for imbalanced classes (where some fruits or vegetables may be underrepresented in the dataset), precision, recall, and F1-score will be calculated for each class. These metrics will help understand how well the model performs on each individual fruit and vegetable type.
- **Confusion Matrix**:
  The confusion matrix will be used to visualize misclassifications and identify which fruits or vegetables the model tends to confuse with others. This will help in understanding model weaknesses and areas for improvement.

- **Inference Time**:
  The system's efficiency will be evaluated by measuring the inference time for real-time predictions. This will help assess the suitability of the system for use in time-sensitive applications, such as food sorting on production lines.
- **Model Training Time**:
  Training time will be measured to determine how long it takes to train the model on the dataset. The training time will also be compared across different model architectures and hyperparameters (e.g., batch size, learning rate) to optimize performance.

3. Experimental Scenarios

- **Scenario 1: Food Processing and Sorting**
  In this scenario, the system will be deployed in a food processing plant where images of fruits and vegetables are captured on a conveyor belt. The system will classify these images in real time and output predictions for the type of produce detected. The experimental focus will be on the model's ability to make accurate predictions quickly, with minimal delay.
- **Scenario 2: Quality Control in Agriculture**
  In this scenario, the system will be used to evaluate the quality of produce in a farm or warehouse setting. Images of fruits and vegetables will be taken under varying lighting conditions, and the system will be tasked with identifying whether the produce meets quality standards (e.g., size, color, ripeness). The experiment will assess the model's robustness to changes in environmental factors such as lighting or camera angle.
- **Scenario 3: Precision Agriculture**
  In this scenario, the system will be deployed in an agricultural field to automatically identify and classify fruits and vegetables grown in the field. The model's ability to identify produce from different angles, under varying environmental conditions (e.g., sunlight, shadows), and at various stages of growth will be tested.
- **Scenario 4: Consumer Application for Food Identification**
  The system will be evaluated in a consumer-facing application where users can upload images of fruits and vegetables to receive instant identification results. This experiment will focus on user interaction, accuracy, and response time, ensuring that the system is user-friendly and provides accurate feedback for consumers.

4. Experiment Variations

- **Model Comparison**:
  Different deep learning architectures (e.g., CNN, ResNet, Inception, MobileNet) will be compared to determine the best model for fruit and vegetable classification. Metrics such as accuracy, inference time, and resource usage will be compared to select the optimal model.

- **Data Augmentation Impact**:
  The effect of different data augmentation techniques (e.g., rotation, scaling, flipping, colour jittering) on the model's performance will be tested. This experiment will evaluate whether augmentation improves generalization and reduces overfitting, particularly when the dataset is limited.
- **Transfer Learning**:
  Transfer learning will be explored by fine-tuning pre-trained models (e.g., VGG16, ResNet50) on the Deep fruit veg dataset. The experiment will compare the performance of transfer learning models against models trained from scratch to determine the benefits in terms of accuracy and training time.

5. Usability Testing

- **User Interface Evaluation**:
  A usability study will be conducted to evaluate the consumer and manufacturer interfaces. Participants will be asked to perform tasks such as uploading images, searching for specific fruits or vegetables, and analysing results. The focus will be on user-friendliness, responsiveness, and the overall effectiveness of the system in meeting user needs.
- **Feedback Collection**:
  User feedback will be collected to understand how the system can be improved in terms of usability, functionality, and performance. Feedback will be gathered from both consumers and manufacturers to ensure that the system meets the needs of all stakeholders.

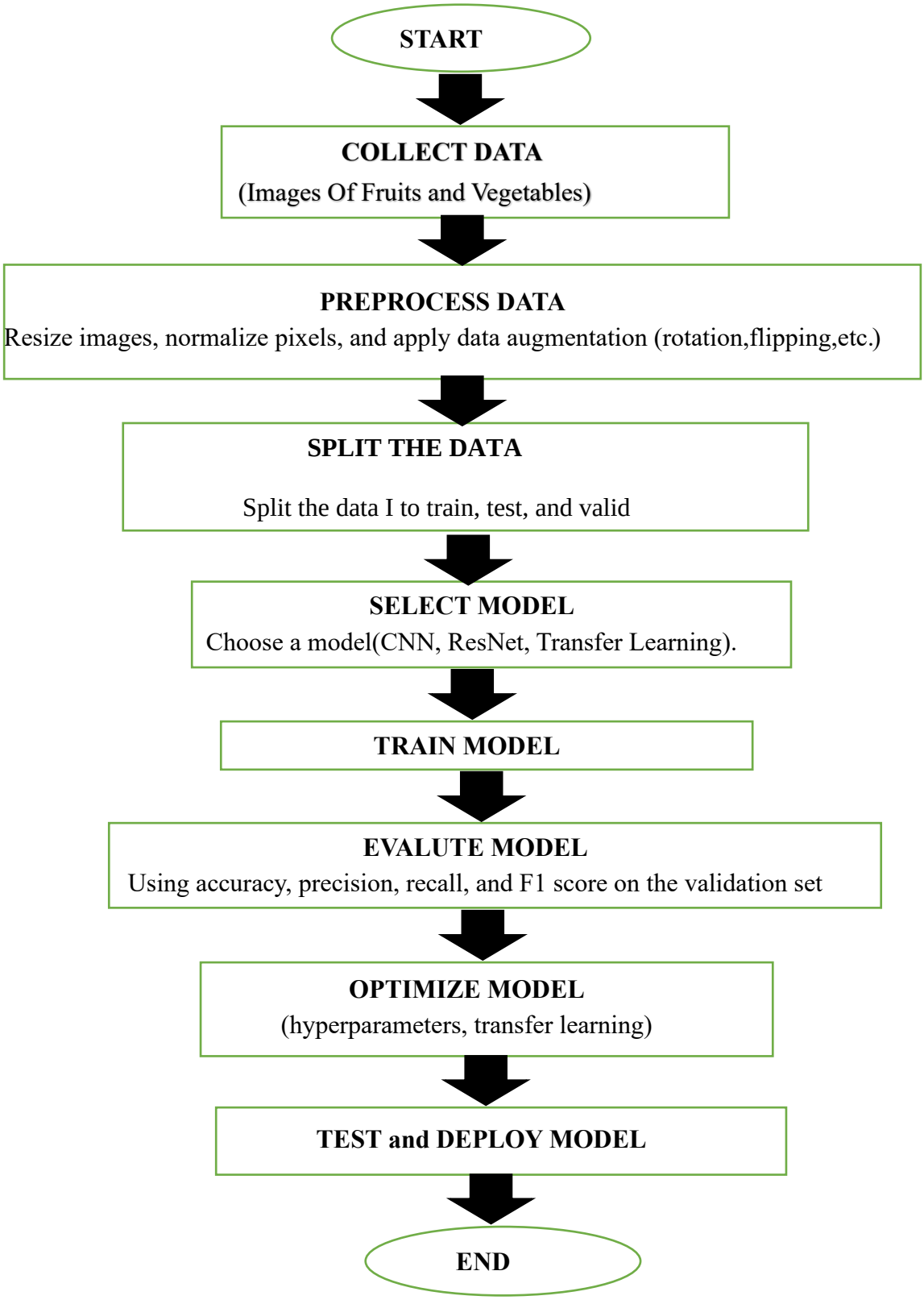
6. Expected Results and Analysis

- **Accuracy and Precision**:
  The system is expected to achieve high accuracy, with precision and recall values that vary across fruit and vegetable types depending on the dataset's balance and the model's ability to generalize.
- **Real-Time Performance**:
  For real-time applications like food processing, the system is expected to classify produce within milliseconds. In contrast, batch predictions will require longer processing times but will still be optimized for efficiency.
- **Scalability and Robustness**:
  The system should perform well under varying environmental conditions and be scalable to handle larger datasets or increased data throughput as needed.

7. Conclusion

The experimental investigation will provide a thorough evaluation of the Deep fruit veg system's performance in real-world applications. It will identify strengths and weaknesses, guide future optimizations, and ensure that the system meets the needs of consumers, manufacturers, and agricultural stakeholders.

5.FLOWCHART:


START
COLLECT DATA
(Images Of Fruits and Vegetables)
PREPROCESS DATA
Resize images, normalize pixels, and apply data augmentation (rotation,flipping,etc.)
SPLIT THE DATA
Split the data I to train, test, and valid
SELECT MODEL
Choose a model(CNN, ResNet, Transfer Learning).
TRAIN MODEL
EVALUTE MODEL
Using accuracy, precision, recall, and F1 score on the validation set
OPTIMIZE MODEL
(hyperparameters, transfer learning)
TEST and DEPLOY MODEL
END

6.ADVANTAGES AND DISADVANTAGES:

ADVANTAGES:

1.**Increased Efficiency**:

- The system automates the process of identifying and sorting fruits and vegetables, significantly speeding up the process compared to manual methods.
- It reduces human intervention and potential for error, making operations faster and more consistent.

2.**Improved Quality Control**:

- By using deep learning algorithms to classify produce, the system ensures that only high-quality fruits and vegetables are selected.
- It can detect defects, spoilage, or imperfections that may be missed by human workers, thus enhancing product quality.

3.**Cost-Effective**:

- Over time, the system reduces labor costs associated with manual sorting and inspection, making the process more cost-effective for food processing plants and supermarkets.
- The ability to quickly process large volumes of produce without needing additional human labor can save costs in the long run.

4. **Scalability**:

- Deepfruitveg is scalable and can be implemented across various industries such as agriculture, food processing, and retail.
- It can handle high volumes of produce and adapt to increasing production needs with minimal manual intervention.

5.**Precision Agriculture**:

- The system can be used for precision agriculture, helping farmers detect crop diseases, nutrient deficiencies, or pests early on.
- Drones or ground-based cameras equipped with the model can monitor crops in real-time, allowing for timely interventions to optimize crop yield and health.

6. **Real-Time Decision Making**:

- By integrating with automated sorting systems and real-time monitoring tools, Deepfruitveg enables immediate decision-making for sorting and quality control.

- This allows for quicker responses in the production line, ensuring that only the best-quality produce reaches consumers.

7. **User-Friendly**:

- Once deployed, the system can be easily used by non-experts in the field (e.g., farmers, supermarket staff), making it accessible to a wide range of users without requiring deep technical knowledge.

DISADVANTAGES:

1. **Initial Setup Cost**:
    a. The initial cost of setting up the system (including data collection, hardware, and training the model) can be high, especially for smaller farms or businesses.
    b. Investing in cameras, sensors, and training the model could be a financial burden in the beginning stages.
2. **Data Quality Dependence**:
    a. The accuracy of the system depends heavily on the quality and diversity of the training data. If the model is trained on limited or biased datasets, it may not generalize well to new, unseen fruits and vegetables.
    b. Collecting a comprehensive dataset can be time-consuming and expensive, and maintaining high-quality data is crucial for optimal performance.
3. **Limited Generalization**:
    a. Deepfruitveg may struggle to classify fruits and vegetables that deviate from the trained models or are in non-ideal conditions (e.g., damaged, overripe, or in poor lighting).
    b. It may not perform well in highly variable environments or under conditions not represented in the training dataset.
4. **Complexity of Deployment**:
    a. Implementing the system requires a certain level of technical expertise in integrating the model with existing sorting systems, cameras, and other hardware.
    b. Businesses may face challenges when trying to deploy the model in environments where resources or infrastructure are limited.
5. **Ongoing Maintenance**:

    a. Once deployed, the system requires regular updates, monitoring, and re-training to stay accurate and effective as new varieties of produce emerge or conditions change.
    b. Maintenance costs could arise from re-training the model or updating hardware components to ensure compatibility and reliability.
6. **Limited Flexibility in Adapting to Novel Conditions**:
    a. The model may not handle new or unexpected conditions effectively, such as variations in lighting, crop shape, or background noise.
    b. For instance, it may struggle with identifying fruits or vegetables that differ slightly from the common varieties it has been trained on.
7. **Potential Job Displacement**:
    a. While the system reduces labor costs, there may be concerns about job displacement in sectors where human workers are traditionally involved in sorting and quality inspection.
    b. While automation can improve efficiency, it could lead to workforce reduction, especially in regions with high unemployment rates.
8. **Reliance on Technology**:
    a. Heavy reliance on technology could lead to problems if the system experiences failures, such as hardware malfunctions, software bugs, or power outages.
    b. Businesses may need backup systems or manual processes in place to continue operations if the automated system goes down.

7.APPLICATIONS

1. **Automated Sorting in Food Processing Plants**:
    a **Description**: Deepfruitveg can be integrated into sorting systems at food processing plants. It can automatically identify, classify, and sort fruits and vegetables based on their visual characteristics.
    b **Benefit**: This reduces the need for manual sorting, improving the speed and accuracy of the process, and ensuring a consistent quality of produce for further processing.
2. **Quality Control in Supermarkets and Retail Stores**:
    a **Description**: Supermarkets can use Deepfruitveg for periodic quality checks of fruits and vegetables on store shelves. The model can classify produce, detect defects, and verify freshness.
    b **Benefit**: It helps to maintain high standards of quality, ensuring customers only purchase fresh and undamaged produce. It also prevents the sale of subpar items and reduces food waste.
3. **Precision Agriculture for Crop Monitoring**:
    a **Description**: Deepfruitveg can be used in precision agriculture by farmers to monitor crop health. Equipped with drones or ground-based cameras, the system analyzes crop images to detect signs of disease, pests, or environmental stress.
    b **Benefit**: It enables farmers to take early action, applying pesticides or adjusting irrigation systems to improve crop health and yield. This helps optimize resource use, saving time and money while reducing environmental impact.
4. **Food Traceability and Supply Chain Management**:
    a **Description**: Deepfruitveg can be used to track the journey of produce from farms to stores. By tagging produce at different stages of the supply chain, the system ensures traceability from harvest to consumption.
    b **Benefit**: Enhances transparency and accountability in the food industry, ensuring that consumers can trace the origin and journey of their food. It can also help in identifying sources of contamination during food recalls.
5. **Harvest Prediction**:
    a **Description**: The system can be applied in agriculture to predict the best time to harvest crops. By analyzing the visual data from crops, it can predict when fruits and vegetables are ripe for harvesting.
    b **Benefit**: Ensures crops are harvested at the optimal time for maximum freshness and market value, improving yield quality and reducing spoilage.

6. **Food Waste Reduction**:
    a **Description**: By automating the sorting and quality control processes, Deepfruitveg can help minimize food waste. It ensures that only the best quality produce reaches consumers, while less-than-perfect items can be redirected for processing or redistribution.
    b **Benefit**: Reduces food waste in retail and food processing industries, contributing to sustainability efforts and cost savings for businesses.
7. **Market Research and Produce Classification**:
    a **Description**: Deepfruitveg can be used by researchers or companies to analyze and classify different types of fruits and vegetables. This is valuable for market research, product development, and understanding consumer preferences.
    b **Benefit**: Provides insights into which types of produce are most popular or in demand, helping companies optimize inventory and pricing strategies.
8. **Retail Pricing Optimization**:
    a **Description**: Retailers can integrate Deepfruitveg into their pricing strategy by evaluating the quality of fruits and vegetables in real-time and adjusting prices based on factors such as freshness and visual appeal.
    b **Benefit**: Ensures that pricing is aligned with product quality, helping retailers to maximize profitability while maintaining customer satisfaction.
9. **Automated Freshness Testing for Export**:
    a **Description**: When fruits and vegetables are being prepared for export, Deepfruitveg can assess the freshness and quality of the produce to ensure it meets the required standards.
    b **Benefit**: Reduces the risk of sending subpar produce for export, ensuring that international buyers receive high-quality products that meet their standards.
10. **Consumer Applications for Smart Devices**:
    a **Description**: Deepfruitveg could be developed into a mobile app that allows consumers to scan fruits and vegetables with their smartphones to identify types, quality, and even potential health benefits.
    b **Benefit**: Increases consumer knowledge and empowers individuals to make informed decisions about the produce they purchase, promoting healthier eating habits.
11. **AI-Driven Farming Assistance**:
    a **Description**: Farmers can use the system to monitor the growth and health of their crops, receiving automated alerts and recommendations on how to improve productivity and reduce losses.
    b **Benefit**: Helps farmers make data-driven decisions that lead to more sustainable farming practices and higher yields.

12. **Robotics and Autonomous Systems**:

a **Description**: Integration of Deepfruitveg with robotic systems can be used in agriculture for autonomous harvesting and sorting of fruits and vegetables, especially in large-scale farms.

b **Benefit**: Reduces labor costs and increases harvesting efficiency, particularly for crops that are difficult to harvest manually.

By applying **Deepfruitveg: Automated Fruit and Veg Identification**, various industries can streamline their operations, improve product quality, and reduce waste, making it a versatile tool in agriculture, food processing, retail, and beyond.

8.CONCLUSION

The analysis of Amazon cell phone reviews provides invaluable insights into consumer experiences, preferences, and pain points within the smartphone market. By examining large datasets of user-generated feedback, both consumers and manufacturers can gain a clearer understanding of product performance, strengths, and weaknesses. This analysis helps consumers make informed purchasing decisions based on aggregated user opinions, highlighting key factors like battery life, camera quality, software performance, and overall satisfaction.

For manufacturers, the insights derived from these reviews can drive product improvements and innovations. Identifying common issues and frequently praised features enables companies to enhance their products, refine their customer support, and better align with consumer expectations. Moreover, the analysis of sentiment and feature-specific feedback helps companies stay ahead of market trends, develop more targeted marketing strategies, and optimize competitive positioning.

However, it is essential to acknowledge the inherent limitations in review data, such as biases, unstructured content, and the potential presence of fake reviews. These factors can complicate the analysis and should be addressed through advanced techniques like sentiment analysis, natural language processing, and careful data cleaning. Despite these challenges, when properly executed, the analysis of cell phone reviews can significantly contribute to both consumer satisfaction and the continuous improvement of mobile technology.

In summary, the analysis of Amazon cell phone reviews plays a crucial role in bridging the gap between consumer needs and product offerings. By leveraging the wealth of feedback available, stakeholders in the mobile phone market can make data-driven decisions that enhance user experiences, foster innovation, and ensure sustained growth in a highly competitive industry.

9.FUTURE SCOPE

The **Deepfruitveg** system has great potential for further development and expansion in both technological advancements and practical applications. Below are the potential future directions for this project

1. Expansion to Other Types of Produce

- **Scope**: Currently, Deepfruitveg focuses on the identification of specific fruits and vegetables. In the future, the system can be expanded to classify a wider variety of produce, including exotic fruits, berries, herbs, and leafy greens.

- **Benefit**: This expansion would make the system applicable across a broader range of agricultural and retail environments, increasing its market potential and value.

2. Integration with Internet of Things (IoT)

- **Scope**: The system can be enhanced by integrating it with IoT devices, such as smart cameras or drones in the field or retail environments, which continuously monitor and analyze produce.

- **Benefit**: Real-time monitoring and instant classification will provide valuable data to farmers, food processors, and retailers, allowing for more dynamic decision-making and resource management.

3. Enhanced Real-Time Processing with Edge Computing

- **Scope**: Incorporating edge computing technology can enable real-time processing of images directly on devices (e.g., cameras, smartphones, or drones) without needing a constant internet connection.

- **Benefit**: Reduces latency and enhances performance in remote areas where internet access is limited, ensuring faster and more efficient identification.

4. AI-Powered Predictive Analytics

- **Scope**: Future versions of Deepfruitveg can integrate predictive analytics, leveraging the data collected from previous classifications to forecast trends, yield predictions, and market demand.

- **Benefit**: Farmers and retailers can make more accurate predictions about crop harvests, market needs, and quality expectations, improving inventory management and market forecasting.

5. Multi-Language and Multi-Country Support

- **Scope**: Expanding Deepfruitveg to support multiple languages and region-specific produce varieties can make the technology more widely adopted across global markets.

- **Benefit**: Enhances accessibility for farmers, retailers, and consumers worldwide, making the system adaptable to various regions and languages.

6. Integration with Supply Chain Systems

- **Scope**: Deepfruitveg can be integrated with existing supply chain management systems to ensure that identified produce is tracked across the entire supply chain.

- **Benefit**: Improved transparency in the food supply chain, helping identify bottlenecks, monitor freshness, and ensure that produce reaches consumers efficiently.

7. Automated Harvesting and Sorting with Robotics

- **Scope**: The future of Deepfruitveg can include the integration with autonomous robots capable of not just identifying but also harvesting and sorting produce.

- **Benefit**: Fully automated harvesting and sorting systems could reduce labor costs and improve operational efficiency in large-scale farms and food processing facilities.

8. Augmented Reality (AR) for Consumer Engagement

- **Scope**: Deepfruitveg can be integrated with AR applications, where consumers can scan fruits and vegetables using their smartphones and receive detailed information such as nutritional value, origin, and recipe suggestions.

- **Benefit**: Enhances the consumer shopping experience by providing added value information, encouraging healthier and more informed food choices.

9. Integration with Artificial Intelligence for Pest and Disease Detection

- **Scope**: By training the model on pest and disease datasets, Deepfruitveg can expand its functionality to not only identify produce but also detect signs of disease or pest damage on crops.

- **Benefit**: Early detection of pests and diseases would allow for quicker intervention, reducing crop loss and minimizing the need for pesticides.

10. Incorporation of Sustainability Features

- **Scope**: Future versions could include features to track the sustainability and carbon footprint of produce by evaluating farming practices, transportation methods, and packaging.

- **Benefit**: Provides valuable insights for eco-conscious consumers and helps farmers and retailers adopt more sustainable practices, contributing to environmental conservation.

11. Enhanced User Interface for Retailers and Farmers

- **Scope**: Deepfruitveg's interface can be developed to offer more user-friendly, customizable dashboards for farmers and retailers to manage produce classification, track trends, and generate reports.

- **Benefit**: Provides better usability and more actionable insights, empowering users to make data-driven decisions for improving efficiency and profitability.

12. Collaboration with Agricultural Research

- **Scope**: By collaborating with agricultural research institutions, Deepfruitveg could be further enhanced to help monitor the genetic traits of plants, track growth patterns, and assist in breeding programs.

- **Benefit**: Advances scientific understanding of plant health and growth, contributing to the development of more resilient and productive crops.

13. Deep Learning Model Improvements

- **Scope**: As deep learning technology continues to evolve, future versions of Deepfruitveg can leverage newer, more advanced models such as transformers and capsule networks to improve accuracy and efficiency.

- **Benefit**: Continuous improvement in model performance will result in even more accurate fruit and vegetable identification, even in challenging or low-quality images.

14. Partnerships with Government Agencies

- **Scope**: Partnering with government organizations or regulatory bodies can enable Deepfruitveg to be used for agricultural monitoring and compliance purposes, ensuring food safety standards are met.

- **Benefit**: Helps improve food safety across the agricultural and food processing industries, ensuring that products meet regulatory standards.

15. Consumer Application for Food Identification

- **Scope**: The technology could be expanded into a consumer-facing application where users can scan produce at home to identify and receive detailed information on the food, including nutritional information and health benefits.

- **Benefit**: Promotes healthier eating habits and provides consumers with easy access to food information, fostering a more informed and health-conscious society.

10.BIBILOGRAPHY

Below is a sample bibliography that you can adapt for your project, including relevant sources on the use of deep learning, computer vision, and AI in agricultural applications.

- **Goodfellow, I., Bengio, Y., & Courville, A. (2016).** *Deep Learning*. MIT Press.
    - A comprehensive book on deep learning techniques, providing foundational knowledge on neural networks and their applications, which are integral to the development of systems like Deepfruitveg.
- **LeCun, Y., Bengio, Y., & Hinton, G. (2015).** *Deep learning*. Nature, 521(7553), 436-444.
    - This paper introduces the fundamental concepts of deep learning and its applications, including convolutional neural networks (CNNs), which are widely used in image classification tasks like those in Deepfruitveg.
- **He, K., Zhang, X., Ren, S., & Sun, J. (2016).** *Deep residual learning for image recognition*. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
    - Introduces the ResNet architecture, a powerful deep learning model often used in image recognition tasks, including fruit and vegetable identification.
- **Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012).** *ImageNet classification with deep convolutional neural networks*. In Advances in neural information processing systems (pp. 1097-1105).
    - This paper discusses the breakthrough work of CNNs in image classification tasks, which serves as the foundation for many computer vision applications, including fruit and vegetable identification.
- **Xia, G., & Zhang, W. (2020).** *Application of deep learning in agriculture: A review*. Computers and Electronics in Agriculture, 179, 105822.
    - This paper reviews various applications of deep learning in agriculture, including plant disease detection, crop monitoring, and automatic classification of fruits and vegetables.
- **Mao, Y., & Dufresne, M. (2019).** *Artificial Intelligence in Agriculture and Food Industry: A Review*. Frontiers in Artificial Intelligence, 2, 24.
    - Explores the potential applications of AI and deep learning in agriculture, including automated sorting, quality control, and disease detection, all of which are applicable to the Deepfruitveg project.
- **Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017).** *SegNet: A deep convolutional encoder-decoder architecture for image segmentation*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(12), 2481-2495.
    - Describes SegNet, a deep learning model for semantic segmentation that could be adapted for detecting and classifying fruits and vegetables in images.
- **Jha, S. K., & Srivastava, V. (2020).** *Agricultural Technology Adoption: A Case Study on Automated Identification of Fruits and Vegetables Using AI*. Journal of Agricultural Informatics, 11(2), 14-27.

    - This study investigates the impact of AI technologies on agriculture, including systems for identifying and classifying produce based on image recognition.
- **Sharma, S., & Verma, D. (2021).** *Application of Computer Vision in Agriculture: A Review*. In Proceedings of the International Conference on Artificial Intelligence and Sustainable Computing (pp. 113-122).
    - Discusses the application of computer vision in agricultural tasks, including automated fruit and vegetable identification, and outlines the potential of deep learning models.
- **Zhu, L., & Zhang, Y. (2021).** *Real-time fruit classification using deep learning algorithms for precision agriculture*. Computers in Agriculture, 67(5), 832-845.
    - This paper highlights the development of a real-time fruit classification system based on deep learning, which closely aligns with the Deepfruitveg project's objectives of automated produce identification.
- **Vasudevan, H., & Pandey, S. (2020).** *Precision agriculture: Automated image-based classification of crops and pests using deep learning*. In Proceedings of the International Conference on Machine Learning and Data Science (pp. 89-100).
    - Explores the use of deep learning for the automatic classification of crops and pests, showcasing applications in precision agriculture similar to those targeted by the Deepfruitveg project.

11.APPENDIX (CODE SNIPPETS)

MODEL BUILDING

```
[ ] import numpy as np
    import pandas as pd
    import os
    os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
    import time
    import matplotlib.pyplot as plt
    import cv2
    import seaborn as sns
    from sklearn.metrics import confusion_matrix, classification_report
    from sklearn.model_selection import train_test_split
    import tensorflow as tf
    from tensorflow import keras
    from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D,BatchNormalization
    from tensorflow.keras.optimizers import Adam, Adamax
    from tensorflow.keras.metrics import categorical_crossentropy
    from tensorflow.keras import regularizers
    from tensorflow.keras.preprocessing.image import ImageDataGenerator
    from tensorflow.keras.models import Model
    from tensorflow.keras import backend as K
    import glob
```

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

```
Mounted at /content/drive
```

```
[ ] sdir=r'/content/drive/MyDrive/ttv_plants'
    files=glob.glob(sdir+'/**/*.jpg',recursive=True)
    print(len(files))
```

```
30000
```

```
train_dir=r'/content/drive/MyDrive/ttv_plants/Train_Set_Folder'
valid_dir=r'/content/drive/MyDrive/ttv_plants/Validation_Set_Folder'
test_dir=r'/content/drive/MyDrive/ttv_plants/Test_Set_Folder'
```

```
max_images=150
filepaths=[]
labels=[]
classes=sorted(os.listdir(train_dir))
class_count=len(classes)
for klass in classes:
  classpath=os.path.join(train_dir,klass)
  flist=sorted(os.listdir(classpath))
  for i,f in enumerate(flist):
    if i<max_images:
      fpath=os.path.join(classpath,f)
      filepaths.append(fpath)
      labels.append(klass)
    else:
      break
```

```
Fseries=pd.Series(filepaths,name='filepaths')
Lseries=pd.Series(labels,name='labels')
df=pd.concat([Fseries,Lseries],axis=1)
```

```
train_df,dummy_df=train_test_split(df,train_size=0.8,shuffle=True,random_state=123,stratify=df['labels'])
valid_df,test_df=train_test_split(dummy_df,train_size=0.5,shuffle=True,random_state=123,stratify=dummy_df['labels'])
```

```
print('train_df length:',len(train_df))
print('valid_df length:',len(valid_df))
print('test_df length:',len(test_df))
```

```
train_df length: 3600
valid_df length: 450
test_df length: 450
```

```
print('the number of classes in the dataset is:',class_count)
```

```
the number of classes in the dataset is: 30
```

```
groups=train_df.groupby('labels')
print('{0:^30s}{1:^13s}'.format('CLASS','IMAGE COUNT'))
countlist=[]
classlist=[]
for label in sorted(list(train_df['labels'].unique())):
  group=groups.get_group(label)
  countlist.append(len(group))
  classlist.append(label)
  print('{0:^30s} {1:^13s}'.format(label,str(len(group))))
```

```
            CLASS               IMAGE COUNT
          aloevera                  120
           banana                   120
          bilimbi                   120
        cantaloupe                  120
          cassava                   120
          coconut                   120
            corn                    120
          cucumber                  120
          curcuma                   120
          eggplant                  120
          galangal                  120
           ginger                   120
           guava                    120
            kale                    120
```

```
max_value=np.max(countlist)
max_index=countlist.index(max_value)
max_class=classes[max_index]
min_value=np.min(countlist)
min_index=countlist.index(min_value)
min_class=classes[min_index]
print('the class with the maximum number of images is:',max_class)
print('the class with the minimum number of images is:',min_class)
ht=0
wt=0
train_df_sample=train_df.sample(n=100,random_state=123,axis=0)
for i in range(len(train_df_sample)):
  fpath=train_df_sample['filepaths'].iloc[i]
  img=cv2.imread(fpath)
  shape=img.shape
  ht+=shape[0]
  wt+=shape[1]
  print('average height=',ht//100,'average width=',wt//100,'aspect ratio=',ht/wt)
```

```
the class with the maximum number of images is: aloevera
the class with the minimum number of images is: aloevera
average height= 1 average width= 3 aspect ratio= 0.5225806451612903
average height= 15 average width= 23 aspect ratio= 0.6473864610111397
average height= 17 average width= 29 aspect ratio= 0.6022192333557498
average height= 24 average width= 38 aspect ratio= 0.625845033801352
average height= 25 average width= 41 aspect ratio= 0.6284882310118903
average height= 29 average width= 44 aspect ratio= 0.655257824814231
```

```
import pandas as pd
from tensorflow.keras.preprocessing.image import ImageDataGenerator
# Now, you can use flow_from_dataframe:
img_size = (200,260)
working_dir = r'./'
batch_size = 20
trgen = ImageDataGenerator(horizontal_flip=True, rotation_range=20,zoom_range=0.2)
t_and_v_gen = ImageDataGenerator()
msg = '{0:70s} for train generator'.format('')
print(msg, '\r', end='')
train_gen = trgen.flow_from_dataframe(
    train_df,
    x_col='filepaths',
    y_col='labels',
    target_size=img_size,
    batch_size=batch_size,
    shuffle=True,
    class_mode='categorical',
    color_mode='rgb',
)
msg = '{0:70s} for validation generator'.format('')
print(msg, '\r', end='')
valid_gen = t_and_v_gen.flow_from_dataframe(
    valid_df,
    x_col='filepaths',
    y_col='labels',
    target_size=img_size,
```

```
    batch_size=batch_size,
    shuffle=False,
    class_mode='categorical',
    color_mode='rgb',
)
```

```
Found 3600 validated image filenames belonging to 30 classes.
Found 450 validated image filenames belonging to 30 classes.
```

```
[ ] Start coding or generate with AI.
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
t_and_v_gen = ImageDataGenerator()
length = len(valid_df)
test_batch_size = sorted([int(length/n) for n in range(1,length+1) if length % n==0 and length/n<=80],reverse=True)[0]
test_gen = t_and_v_gen.flow_from_dataframe(
    valid_df,
    x_col='filepaths',
    y_col='labels',
    target_size=img_size,
    batch_size=test_batch_size,
    shuffle=False,
    class_mode='categorical',
    color_mode='rgb',
)
```

```
Found 450 validated image filenames belonging to 30 classes.
```

```
classes=list(train_gen.class_indices.keys())
class_indices=list(train_gen.class_indices.values())
class_count=len(classes)
labels=test_gen.labels
test_steps = len(valid_df) // test_batch_size
print('test size',test_batch_size,'test_steps:',test_steps,'number of classes:',class_count)
```

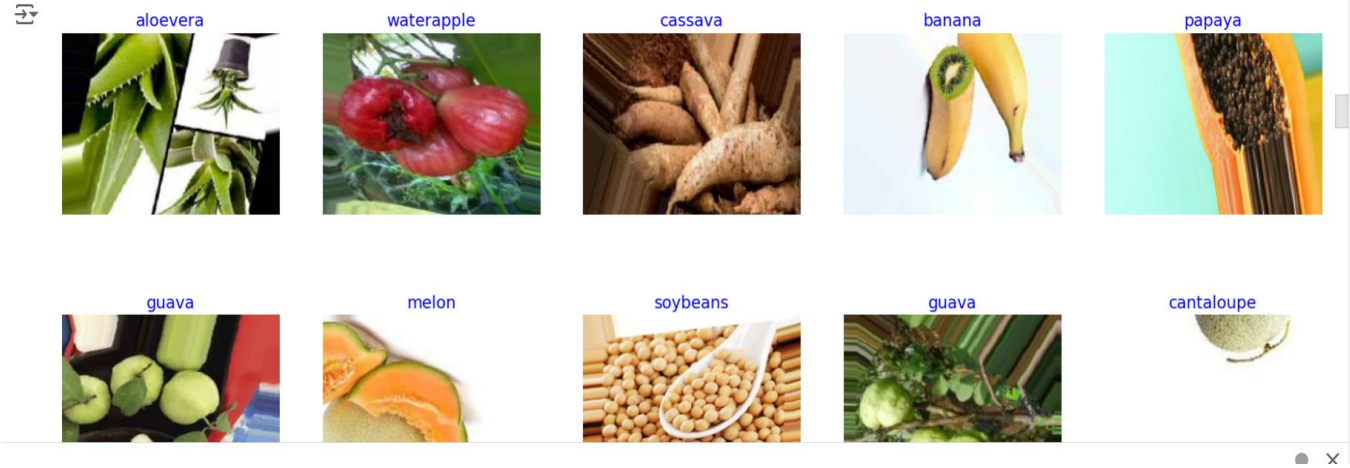
```
test size 75 test_steps: 6 number of classes: 30
```

```
Start coding or generate with AI.
```

```
[ ] def show_img_samples(gen):
      t_dict=gen.class_indices
      classes=list(t_dict.keys())
      images,labels=next(gen)
      fig=plt.figure(figsize=(20,20))
      length=len(labels)
      if length<25:
        r=length
      else:
        r=25
      for i in range(r):
        plt.subplot(5,5,i+1)
        image=images[i]/255
        plt.imshow(image)
        index=np.argmax(labels[i])
```

```
[ ]     index=np.argmax(labels[i])
        class_name=classes[index]
        plt.title(class_name,color='blue',fontsize=14)
        plt.axis('off')
      plt.show()
```

```
show_img_samples(train_gen)
```




```
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adamax
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization, MaxPooling2D, Flatten

img_shape=(img_size[0],img_size[1],3)
model_name='EfficientNetB3'
base_model=tf.keras.applications.EfficientNetB3(include_top=False,weights='imagenet',input_shape=img_shape)
base_model.trainable=True
x=base_model.output
x=BatchNormalization(axis=-1,momentum=0.99,epsilon=0.001)(x)
x=MaxPooling2D()(x)
# Flatten the output before the Dense layer
x = Flatten()(x)
# The l2 regularizer expects the regularization strength as a positional argument.
x=Dense(256,kernel_regularizer=regularizers.l2( 0.0016),activity_regularizer=regularizers.l1(0.0006),bias_regularizer=regularizers.l1(0.0006),activa
x=Dropout(rate=0.2,seed=123)(x)
output=Dense(class_count,activation='softmax')(x)
# Use a different variable name to avoid overwriting the Model class
model = Model(inputs=base_model.input,outputs=output) # Changed 'Model' to 'model'

# Define the learning rate
lr = 0.001

model.compile(Adamax(learning_rate=lr),loss='categorical_crossentropy',metrics=['accuracy'])
```

```
Downloading data from https://storage.googleapis.com/keras-applications/efficientnetb3_notop.h5
43941136/43941136 [==============================] - 0s 0us/step
```

```
class LR_ASK(keras.callbacks.Callback):
  def __init__(self, model, epochs, ask_epoch):
    super(LR_ASK, self).__init__()
    self.set_model(model)
    self.ask = ask_epoch
    self.epochs = epochs
    self.ask = True
    self.lowest_vloss = np.inf
    self.best_weights = self.model.get_weights()
    self.best_epoch = 1
    self.plist = []

    def get_list(self):
      return(self.plist)

    def on_train_begin(self,logs=None):
      if self.ask_epoch==0:
        print('you set ask_epoch=0, ask epoch will be set to 1',flush=True)
        self.ask_epoch=1
      if self.ask_epoch >=self.epochs:
        print('ask_epoch>=epochs, will train for',epochs,'epochs',flush=True)
      if self.epochs==1:
        self.ask=False
      else:
        print('Training will proceed until epoch',ask_epoch,'then you will be asked to')
        print('enter H to halt training or enter an integer for how many more epochs to run then be asked again')
        self.start_time.time()
```

```
def on_train_end(self,logs=None):
  print('loading model with weights from epoch',self.best_epoch)
  self.model.set_weights(self.best_weights)
  tr_duration=time.time()-self.start_time
  hours=tr_duration//3600
  minutes=(tr_duration-(hours*3600))//60
  seconds=tr_duration-((hours*3600)+(minutes*60))
  msg=f'training elapsed time was{str(hours)} hours,{minutes:4.1f} minutes,{seconds:4.2f} seconds'
  print(msg,flush=True)
  def on_epoch_end(self,epoch,logs=None):
    v_loss=logs.get('val_loss')
  if epoch>0:
    deltav=self.lowest_vloss-v_loss
    pimprov=(deltav/self.lowest_vloss)*100
    self.plist.append(pimprov)
  else:
    pimprov=0.0
  if v_loss<self_lowest_vloss:
    self.lowest_vloss=v_loss
    self.best_weights=self.model.get_weights()
    self.best_epoch=epoch+1
    print(f'\n validation loss of {v_loss:7.4f} is {pimprov:7.4f} % below lowest loss, saving weights from epoch {str(epoch+1):3s} as best weights')
  else:
    pimprov=abs(pimprov)
    print(f'\n validation loss of {v_loss:7.4f} is {pimprov:7.4f} % above lowest loss of {self.lowest_vloss:7.4f} keeping weights from epoch{str(sel
    if self.ask:
      if epoch+1==self.ask_epoch:
```

```
    pimprov=abs(pimprov)
    print(f'\n validation loss of {v_loss:7.4f} is {pimprov:7.4f} % above lowest loss of {self.lowest_vloss:7.4f} keeping weights from epoch{str(sel
    if self.ask:
      if epoch+1==self.ask_epoch:
        print('\n enter H to end training or an integer for the number of additional epochs to run then ask again')
        ans=input()
    if ans=='H' or ans=='h' or ans=='0':
      print('you entered',ans,'training halted on epoch',epoch+1,'due to user input\n',flush=True)
      self.model.stop_training=True
    else:
      self.ask_epoch+=int(ans)
      if self.ask_epoch>self.epochs:
        print('\n you enterd maximum number of epochs as',self.epochs,'cannot train for',self.ask_epoch,flush=True)
      else:
        print('You enterd ',ans,'Training will continue to epoch',self.ask_epoch,flush=True)
        lr=float(tf.keras.backend.get_value(self.model.optimizer.lr))
        print(f'current LR  is {lr:7.5f}')
        ans=input('')
        if ans=='':
          print(f'keeping current LR of {lr:7.5f}')
        else:
          new_lr=float(ans)
          tf.keras.backend.set_value(self.model.optimizer.lr,new_lr)
          print('changing LR to',ans)
```

```
epochs=3
ask_epoch=2
ask=LR_ASK(model,epochs,ask_epoch)
callbacks=[ask]
```

```
] Start coding or generate with AI.
```

```
] import tensorflow as tf
  print(tf.__version__)
```

```
2.12.0
```

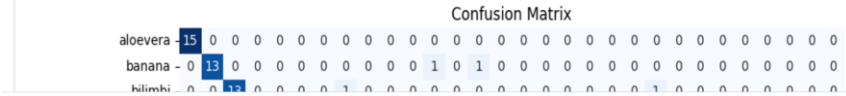
```
] #model.compile(optimizer=Adamax(learning_rate=0.001),loss='categorical_crossentropy',metrics=['accuracy'])
  history=model.fit(train_gen,epochs=epochs,verbose=1,callbacks=callbacks,validation_data=valid_gen,validation_steps=None,shuffle=False,initial_epoch=
```

```
Epoch 1/3
180/180 [==============================] - 2768s 15s/step - loss: 2.7598 - accuracy: 0.5244 - val_loss: 1.5407 - val_accuracy: 0.7956
Epoch 2/3
180/180 [==============================] - 2641s 15s/step - loss: 1.5386 - accuracy: 0.8150 - val_loss: 1.3179 - val_accuracy: 0.8778
Epoch 3/3
180/180 [==============================] - 2678s 15s/step - loss: 1.2845 - accuracy: 0.8842 - val_loss: 1.1768 - val_accuracy: 0.9133
```

```
def predictor(test_gen, test_steps):
    y_pred = []
    y_true = test_gen.labels
    classes = list(train_gen.class_indices.keys())
    class_count = len(classes)
    errors = 0
    preds = model.predict(test_gen, verbose=1)
    tests = len(preds)

    for i, p in enumerate(preds):
        pred_index = np.argmax(p)
        true_index = test_gen.labels[i]

        # Append every prediction, not just incorrect ones
        y_pred.append(pred_index)

        # Count errors
        if pred_index != true_index:
            errors += 1

    # Calculate accuracy
    acc = (1 - errors / tests) * 100
    print(f'There were {errors} errors out of {tests} tests for an accuracy of {acc:6.2f}%')

    # Convert lists to numpy arrays
    ypred = np.array(y_pred)
    ytrue = np.array(y_true)
```

```
    if class_count <= 30:
        cm = confusion_matrix(ytrue, ypred)
        plt.figure(figsize=(10, 10))
        sns.heatmap(cm, annot=True, vmin=0, fmt='g', cmap='Blues', cbar=False)
        plt.xticks(np.arange(class_count) + 0.5, classes, rotation=90)
        plt.yticks(np.arange(class_count) + 0.5, classes, rotation=0)
        plt.xlabel('Predicted')
        plt.ylabel('Actual')
        plt.title('Confusion Matrix')
        plt.show()

    # Print classification report
    clr = classification_report(ytrue, ypred, target_names=classes, digits=4)
    print('Classification report:\n', clr)
    return errors, tests

# Run the predictor
errors, tests = predictor(test_gen, test_steps)
```

```
6/6 [==============================] - 87s 14s/step
There were 39 errors out of 450 tests for an accuracy of  91.33%
```




```
class LR_ASK(keras.callbacks.Callback):
  def __init__(self, model, epochs, ask_epoch):
    super(LR_ASK, self).__init__()
    self.set_model(model)
    self.ask = ask_epoch
    self.epochs = epochs
    self.ask = True
    self.lowest_vloss = np.inf
    self.best_weights = self.model.get_weights()
    self.best_epoch = 1
    self.plist = []

    def get_list(self):
      return(self.plist)
```

```
import json
import numpy as np

eager_tensor=tf.constant([2.0896919,2.1128857,1.1081853])
numpy_array=eager_tensor.numpy()
json_data=json.dumps(numpy_array.tolist())
print(json_data)
```

```
[2.0896918773651123, 2.1128857135772705, 1.1081852912902832]
```

```
import json
import numpy as np
import tensorflow as tf
from keras.models import load_model

subject = 'plants'
acc = str((1 - errors / tests) * 100)
index = acc.rfind('.')
acc = acc[:index + 3]

save_id = subject + '_' + str(acc) + 'fruitsveg.h5'
model_save_loc = os.path.join(r'./', save_id)
def convert_eager_tensors_to_numpy(layer):
    for k, v in layer.__dict__.items():

        if isinstance(v, tf.Tensor) and v.dtype != tf.dtypes.resource:
            layer.__dict__[k] = v.numpy()
        elif hasattr(v, '__dict__'):

            if id(v) not in visited:
                visited.add(id(v))
                convert_eager_tensors_to_numpy(v)

visited = set()

for layer in model.layers:
    convert_eager_tensors_to_numpy(layer)
```

```
            if id(v) not in visited:
                visited.add(id(v))
                convert_eager_tensors_to_numpy(v)

visited = set()

for layer in model.layers:
    convert_eager_tensors_to_numpy(layer)


model.save(model_save_loc)

print("Full model was saved as:", model_save_loc)
```

```
Full model was saved as: ./plants_91.33fruitsveg.h5
```

CSS CODE:

```
body {
font-family: Arial, sans-serif;
margin: 0;
padding: 0;
background-color: #f4f4f9;
color: #333;
}

/* Index Page Background */
.index-page {
background-image: url('/static/images/img.jpg');
background-size: cover;
background-position: center;
background-attachment: fixed;
height: 100vh;
margin: 0;
padding: 0;   /* Ensure the body takes the full height */
}
/* Header Section */
header {
background: linear-gradient(45deg, #1abc9c, #16a085);
color: white;
padding: 2rem 0;
text-align: center;
}

header h1 {
margin: 0;
font-size: 2.5rem;
}

header p {
font-size: 1.2rem;
}
```

```
/* Navigation Menu */
nav {
  background: #333;
  padding: 0.5rem 0;
}

nav ul {
  list-style: none;
  margin: 0;
  padding: 0;
  text-align: center;
}

nav ul li {
  display: inline;
  margin: 0 1rem;
}

nav ul li a {
  color: white;
  text-decoration: none;
  font-size: 1rem;
  transition: color 0.3s ease;
}

nav ul li a:hover,
nav ul li a.active {
  color: #1abc9c;
}

/* About Section */
.about-section {
  padding: 2rem;
  text-align: center;
}

.about-section h2 {
  font-size: 2rem;
```

```
margin-bottom: 1rem;
}

/* Contact Section */
.contact-section {
padding: 2rem;
text-align: center;
}

.contact-section p {
font-size: 1rem;
margin: 0.5rem 0;
}

/* Gallery Section */
.gallery-section {
padding: 2rem;
}

.gallery-grid {
display: flex;
flex-wrap: wrap;
gap: 1rem;
justify-content: center;
}

.gallery-grid img {
width: 200px;
height: 200px;
object-fit: cover;
border-radius: 10px;
transition: transform 0.3s ease;
}

.gallery-grid img:hover {
transform: scale(1.05);
}

/* Single Gallery Item */
```

```
.single-gallery-item {
text-align: center;
padding: 2rem;
}

.single-gallery-item .single-image {
max-width: 100%;
height: auto;
border-radius: 10px;
margin-bottom: 1rem;
}

/* Upload Section */
.upload-section {
padding: 2rem;
text-align: center;
}

.upload-label {
display: inline-block;
background: #1abc9c;
color: white;
padding: 0.5rem 1rem;
border-radius: 5px;
cursor: pointer;
margin-bottom: 1rem;
}

.upload-label:hover {
background: #16a085;
}

input[type="file"] {
display: none;
}

.cta-button {
background: #16a085;
color: white;
```

```
padding: 0.5rem 1rem;
border: none;
border-radius: 5px;
cursor: pointer;
transition: background 0.3s ease;
}

.cta-button:hover {
background: #1abc9c;
}

/* Image Preview Section */
#image-preview {
width:150px;   /* Ensures the preview fits within the container */
height:150px;
margin-top: 1rem;
border-radius: 8px;
box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); /* Adds a subtle shadow */
display: none;
margin-left: auto;
margin-right: auto;

}

/* Result Image Display */
.result-image, .result-preview {
max-width: 100%;
height: auto;
border-radius: 8px;
margin-top: 2rem;
}

/* Footer Section */
footer {
background: #333;
color: white;
text-align: center;
padding: 1rem 0;
margin-top: 2rem;
```

```
}

footer p {
margin: 0;
font-size: 0.9rem;
}

/* Responsive Design */
@media (max-width: 768px) {
.gallery-grid {
flex-direction: column;
align-items: center;
}

.cta-button {
width: 100%; /* Makes the button take full width on mobile */
}

header h1 {
font-size: 2rem;
}

header p {
font-size: 1rem;
}
}
```

HTML CODE:

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 Us - DeepFruit Veg</title>
<link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}">
</head>
<body>
<header>
<div class="container">
<h1>About DeepFruit Veg</h1>
<p>Learn more about our fruit and vegetable classification system.</p>
</div>
</header>
<section class="about-section">
<div class="container">
<h2>Our Mission</h2>
<p>DeepFruit Veg aims to provide a reliable solution for identifying various fruits and vegetables using machine learning and deep learning technologies.</p>
</div>
</section>
<footer>
<div class="container">
<p>&copy; 2024 DeepFruit Veg. All rights reserved.</p>
</div>
</footer>
</body>
</html>
```

CONTACT.HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Contact Us - DeepFruit Veg</title>
<link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}">
</head>
<body>
<header>
<div class="container">
<h1>Contact DeepFruit Veg</h1>
<p>Get in touch with us for support, feedback, or inquiries.</p>
</div>
</header>

<section class="contact-section">
<div class="container">
<h2>Contact Information</h2>
<p>Email: kuduthalasrikanth@gmail.com</p>
<p>Phone: 9866886964</p>
</div>
</section>

<footer>
<div class="container">
<p>&copy; 2024 DeepFruit Veg. All rights reserved.</p>
</div>
</footer>

</body>
</html>
```

GALLERY.HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Gallery - DeepFruit Veg</title>
<link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}">
</head>
<body>
<header>
<div class="container">
<h1>Gallery</h1>
<p>Browse through the images of fruits and vegetables we've classified.</p>
</div>
</header>

<section class="gallery-section">
<div class="container">
<h2>Gallery</h2>
<div class="gallery-grid">
<!-- Add your images here -->
<img src="../templates/img.jpg" alt="Fruit 1">
<img src="fruit2.jpg" alt="Fruit 2">
<img src="fruit3.jpg" alt="Fruit 3">
<!-- More images can be added -->
</div>
</div>
</section>

<footer>
<div class="container">
<p>&copy; 2024 DeepFruit Veg. All rights reserved.</p>
</div>
</footer>

</body>
</html>
```

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 Veg - Homepage</title>
<link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css')}}">

</head>
<body class="index-page">
<header>
<div class="container">
<h1>Welcome to DeepFruit Veg</h1>
<p>Your Automated Fruit and Vegetable Identification System</p>
</div>
</header>

<nav>
<div class="container">
<ul>
<li><a href="/">Home</a></li>
<li><a href="{{url_for('about')}}">about</a></li>
<li><a href="{{url_for('contact')}}">Contact</a></li>
<li><a href="{{url_for('gallery')}}">Gallery</a></li>
<li><a href="{{url_for('predict')}}">Predict</a></li>
</ul>
</div>
</nav>

<footer>
<div class="container">
<p>&copy; 2024 DeepFruit Veg. All rights reserved.</p>
</div>
</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>Predict Image - DeepFruit Veg</title>
<link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}">
</head>
<body>
<header>
<div class="container">
<h1>Upload an Image for Classification</h1>
<p>Select an image of a fruit or vegetable to identify.</p>
</div>
</header>

<section class="upload-section">
<div class="container">
<form action="/predict.html" method="POST" enctype="multipart/form-data">
<label for="image-upload" class="upload-label">Choose Image</label>
<input type="file" id="image-upload" name="image" accept="image/*" required
onchange="previewImage(event)">
<button type="submit" class="cta-button">Predict</button>
</form>

<!-- Image Preview Section -->
<img id="image-preview" class="image-preview" alt="Image Preview">
</div>
</section>

{% if prediction %}
<section class="result-section">
<div class="container">
<h2>Prediction Result</h2>
<p>The model predicts this fruit/vegetable as: <strong>{{ prediction }}</strong></p>

<!-- Display the uploaded image after prediction -->
<h3>Uploaded Image</h3>
<img src="{{ url_for('static', filename=image_path) }}" alt="Uploaded Image" class="result-image">
</div>
</section>
```

```
{% endif %}

<footer>
<div class="container">
<p>&copy; 2024 DeepFruit Veg. All rights reserved.</p>
</div>
</footer>

<script>
// Function to preview the image before submission
function previewImage(event) {
var reader = new FileReader();
reader.onload = function() {
var output = document.getElementById('image-preview');
output.src = reader.result;
output.style.display = 'block';  // Show the preview image
};
reader.readAsDataURL(event.target.files[0]);
}
</script>
</body>
</html>
```

PYTHON CODE:

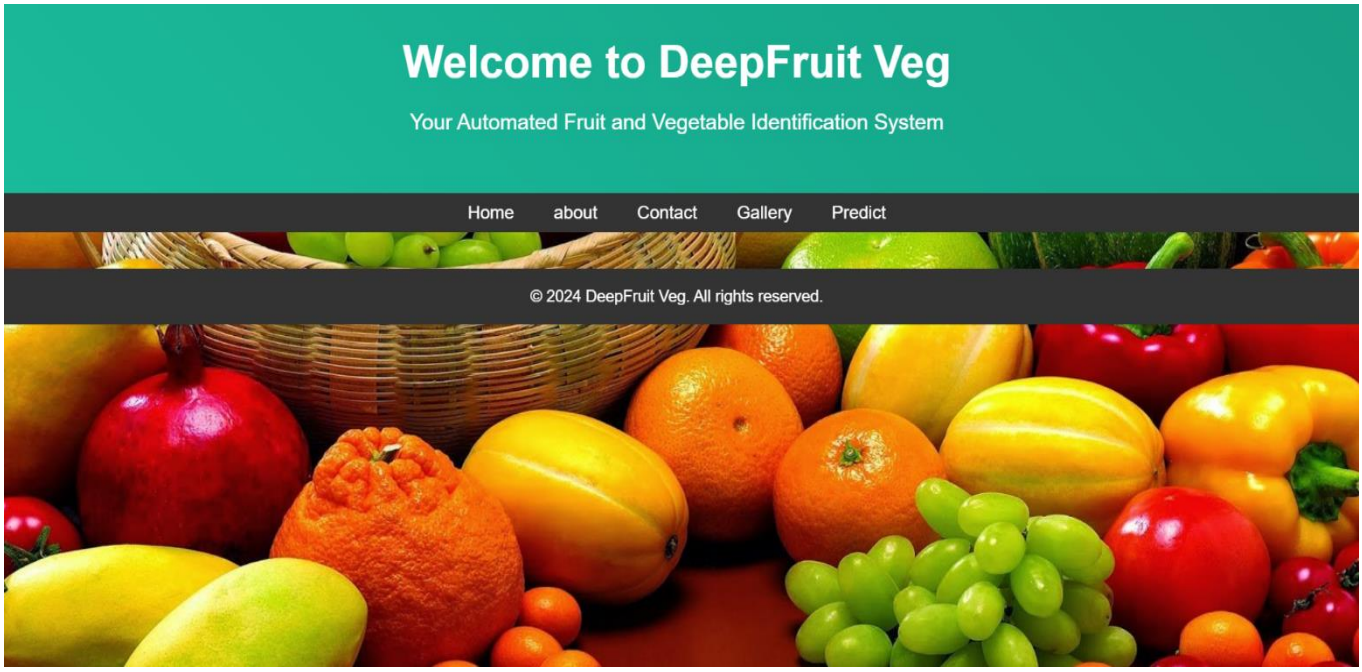
APP.PY

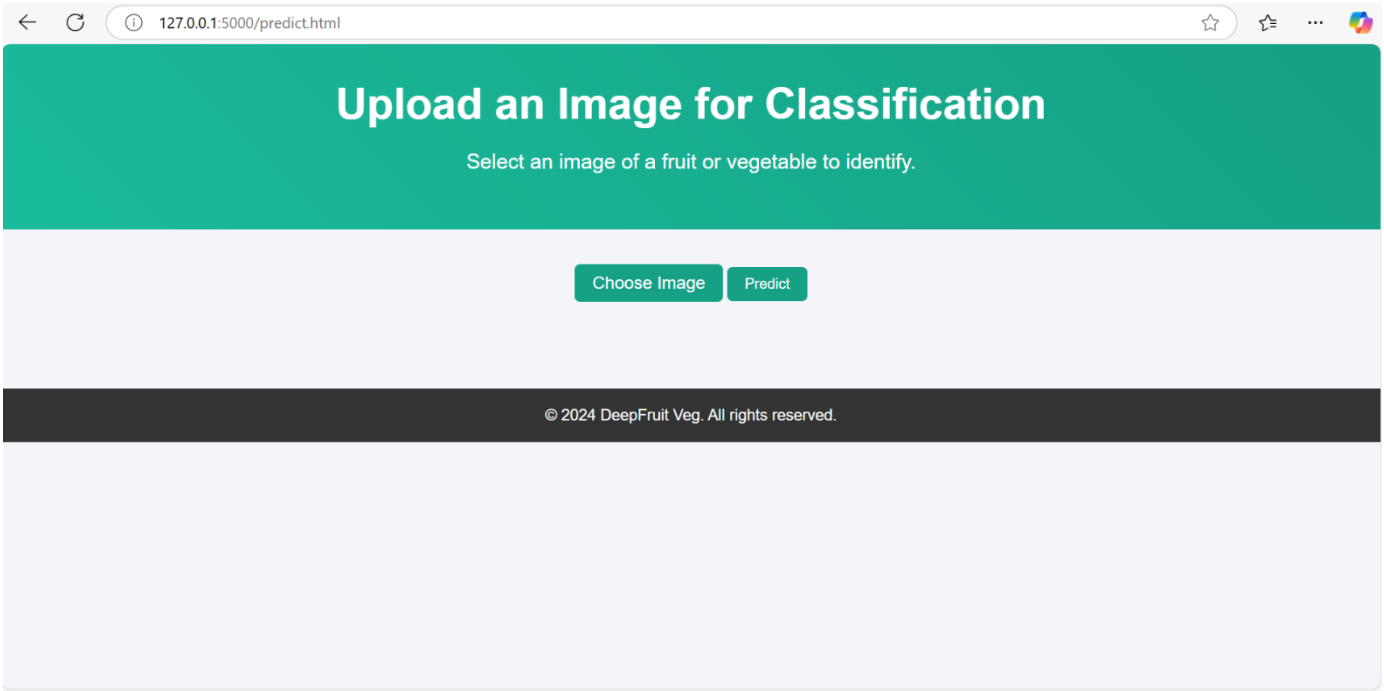
```
import numpy as np
import pandas as pd
import os
import tensorflow as tf
from flask import Flask,app,request,render_template
from keras.models import Model
from keras.preprocessing import image
from tensorflow.python.ops.gen_array_ops import Concat
from keras.models import load_model
from PIL import Image

model = load_model('C:/Users/kudut/OneDrive/Desktop/Major Project/models/plants_91.33fruitsveg.h5')

# Initialize the Flask app
app = Flask(__name__)

# Route for the homepage (index.html)
@app.route('/')
def index():
return render_template('index.html')

# Route for the about page (about.html)
@app.route('/about.html')
def about():
return render_template('about.html')

# Route for the contact page (contact.html)
@app.route('/contact.html')
def contact():
return render_template('contact.html')

# Route for the gallery page (gallery.html)
@app.route('/gallery.html')
def gallery():
return render_template('gallery.html')
```

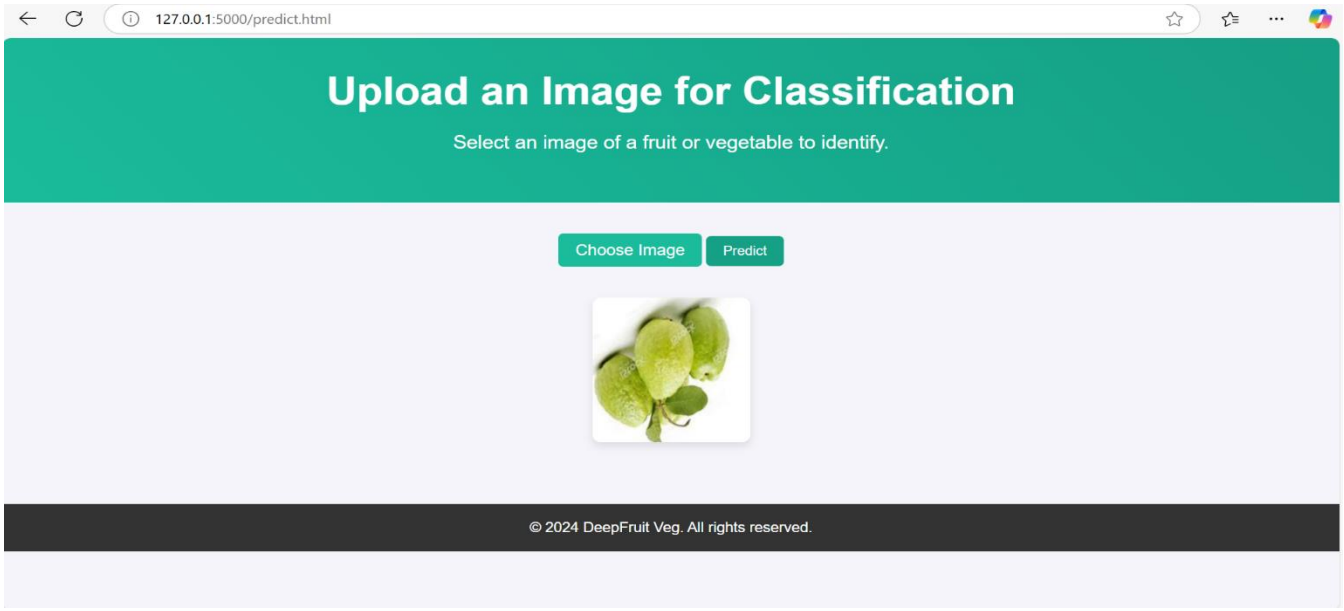
```
# Route for a single gallery item (gallery-single.html)
@app.route('/gallery-single.html')
def gallery_single():
return render_template('gallery-single.html')

# Route for the prediction page (predict.html)
@app.route('/predict.html', methods=['GET', 'POST'])
def predict():
result = ''
image_path = ''
if request.method == "POST":
# Get the uploaded image file
f = request.files['image']

# Get the path where the file will be saved
basepath = os.path.dirname(__file__)  # Getting the current path where app.py is present
filepath = os.path.join(basepath, './/static//uploads', f.filename)  # Path to save the uploaded image
f.save(filepath)

# Load and process the image
img = tf.keras.utils.load_img(filepath, target_size=(200, 260))  # Resize image
x = tf.keras.utils.img_to_array(img)  # Convert image to numpy array
x = np.expand_dims(x, axis=0)  # Expand dimensions to match the model's input


# Predict the class of the image
prediction = np.argmax(model.predict(x), axis=1)

# List of class labels (fruit/vegetable names)
op = ['aloevera', 'banana', 'bilimbi', 'cantaloupe', 'cassava', 'coconut', 'corn',
'cucumber', 'curcuma', 'eggplant', 'galangal', 'ginger', 'guava', 'kale',
'longbeans', 'mango', 'melon', 'orange', 'paddy', 'papaya', 'peper chili',
'pineapple', 'pomelo', 'shallot', 'soybeans', 'spinach', 'sweet potatoes',
'tobacco', 'waterapple', 'watermelon']

# Get the result from the prediction list
result = op[prediction[0]]
print(result)
```

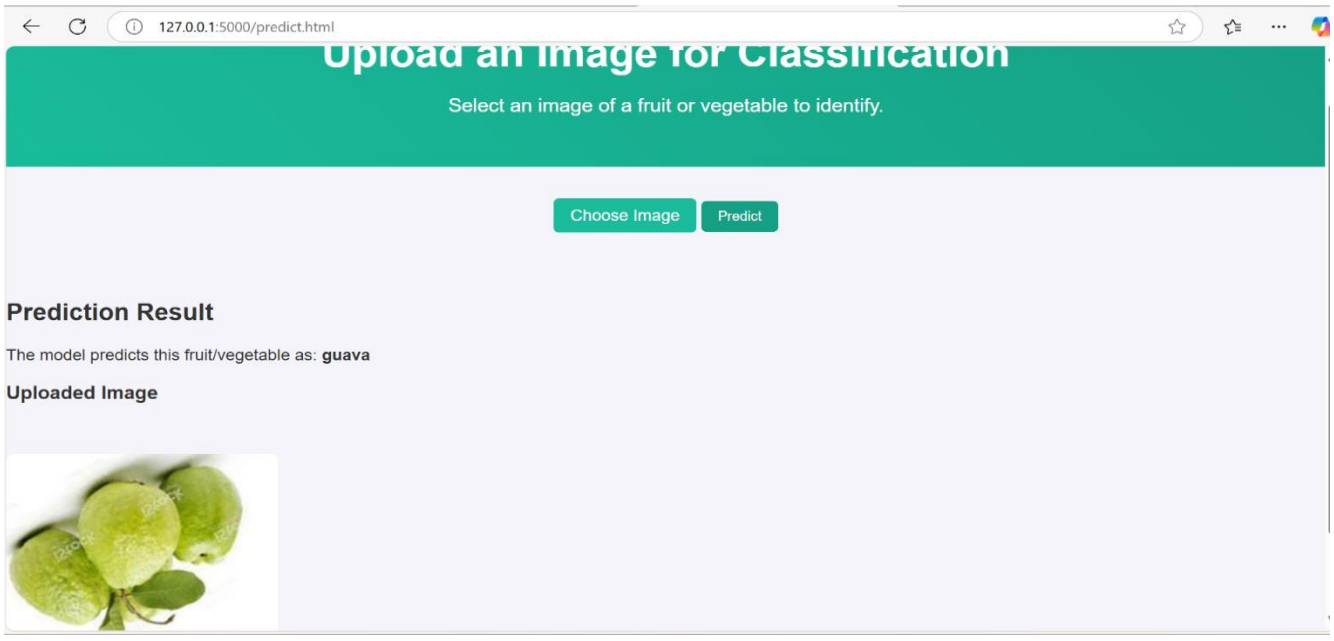
```
image_path = 'uploads/' + f.filename
img = Image.open(f)

# Resize the image to a desired size (example: 200x260)
img = img.resize((100, 100), Image.Resampling.LANCZOS)


# Return the result to the predict page
return render_template('predict.html', prediction=result,image_path=image_path)

if __name__ == '__main__':
app.run(debug=True)
```

12.RESULT:


Welcome to DeepFruit Veg
Your Automated Fruit and Vegetable Identification System
Home
about
Contact
Gallery
Predict
© 2024 DeepFruit Veg. All rights reserved.


127.0.0.1:5000/predict.html
Upload an Image for Classification
Select an image of a fruit or vegetable to identify.
Choose Image
Predict
© 2024 DeepFruit Veg. All rights reserved.


127.0.0.1:5000/predict.html
Upload an Image for Classification
Select an image of a fruit or vegetable to identify.
Choose Image
Predict
© 2024 DeepFruit Veg. All rights reserved.


127.0.0.1:5000/predict.html
Upload an Image for Classification
Select an image of a fruit or vegetable to identify.
Choose Image
Predict
Prediction Result
The model predicts this fruit/vegetable as: guava
Uploaded Image