Project Document

Project Title:

Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables

Team Name:

FreshDetect Squad

Team Members:

B.Veera Mallika,(team leader)

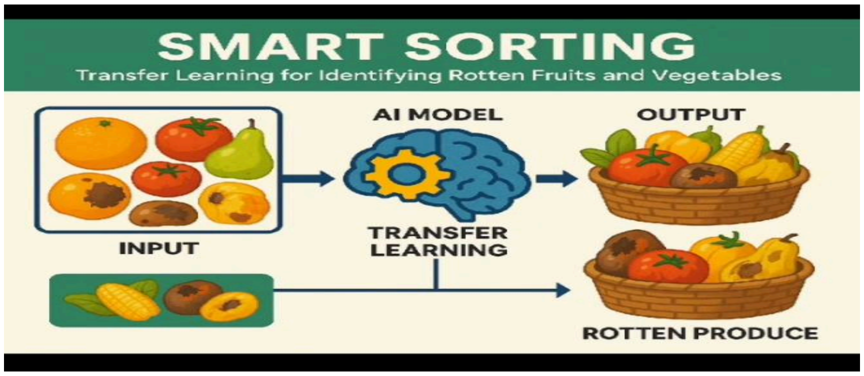
C.Pushpavathi,

B.Lalithama,

B.Sai Chandana,

B.Maheshwari.

Introduction



In today's rapidly advancing world, technology plays a crucial role in improving food quality and safety. One of the major challenges in the food supply chain is the identification and removal of rotten fruits and vegetables, which can lead to health risks and food wastage. This project, titled "Smart Sorting of Fresh and Rotten Fruits and Vegetables using Transfer Learning", presents an intelligent system that leverages the power of machine learning to automatically detect and classify fresh and rotten produce.

By using transfer learning techniques and deep learning models, this system is trained to recognize subtle differences in fruit and vegetable images, enabling fast, reliable, and accurate sorting. The goal of the project is to minimize human error, enhance efficiency in the sorting process, and contribute to smarter agricultural and retail practices.

Project Document

This include few phases like :

- **Brainstorming & Ideation**
- **Requirement Analysis**
- **Project Design**
- **Project Planning (Agile Methodologies)**
- **Project Development**
- **Functional & Performance Testing :**

Phase 1: Brainstorming & Ideation

Objective of this Phase:

The primary goal of this phase is to explore innovative ways to **determine the freshness of fruits and vegetables without any human involvement**. This foundational idea serves as the basis for developing an intelligent, automated solution using artificial intelligence (AI).

Core Idea:

We aim to build a **novel AI-based system** capable of detecting the freshness level of fruits and vegetables by analyzing their images. The system should function autonomously, providing real-time insights with **no manual inspection required**.

Problem Statement:

Traditional methods of checking the freshness of produce are:

- **Manual and time-consuming**
- **Prone to human error and inconsistency**
- A leading cause of **food wastage**, **customer dissatisfaction**, and **economic losses**, especially for farmers and retailers

---

Proposed Solution:

To address this issue, we propose to design and develop an **automated image analysis system**. This system will:

- Capture and analyze images of fruits and vegetables
- Classify them as **fresh** or **rotten** based on visual cues
- Operate with high accuracy using AI and computer vision

- Eliminate the need for manual checking, thereby ensuring faster and more reliable sorting

Target Users and Expected Outtcome:

The target users for this system include **farmers and agricultural workers**, who can benefit from on-site sorting; **supply chain managers**, who require efficient quality checks during transportation; **supermarket and grocery store employees**, responsible for shelf management; and **cold storage or warehouse operators**, who handle bulk produce storage. The expected outcome is a **functional, image-based sorting system** capable of accurately identifying and classifying fruits and vegetables as **fresh or rotten**. This intelligent tool can be effectively deployed in both **agricultural** and **retail environments** to significantly reduce food waste and enhance overall **operational efficiency** across the supply chain.

Phase 2: Requirement Analysis

After thoroughly analyzing the core idea behind the project, **Requirement Analysis** becomes the next critical phase. This stage focuses on determining both the **technical tools** and the **functional capabilities** necessary for successfully developing the AI-based freshness detection system. By clearly outlining these requirements, we ensure that the implementation process is smooth, structured, and goal-oriented.

In terms of **technical requirements**, the project is developed using the **Python programming language** due to its simplicity and strong ecosystem for machine learning and image processing tasks. Several important libraries are utilized, each of which can be installed using pip, the Python package installer:

- **TensorFlow** – used for building and deploying machine learning models

```
pip install tensorflow
```

- **Keras** – a high-level API for neural network implementation (now integrated with TensorFlow)
  (Note: Installing TensorFlow will also install Keras)

OpenCV – for image processing and computer vision tasks

```
pip install opencv-python
```

- **NumPy** – supports efficient numerical computations and array manipulation

```
pip install numpy
```

- **Matplotlib** – used for visualizing training progress and displaying images pip install matplotlib

- **Pillow (PIL)** – for opening, manipulating, and converting images

pip install pillow

These libraries form the **technical backbone** of the system, enabling effective development, real-time inference, and image handling throughout the pipeline.

On the **functional side**, the system must:

- Support **real-time classification**
- Include a **user interface** for interaction and display of results
- Perform **image preprocessing** such as resizing and normalization
- Handle **batch datasets** for model training and testing

By specifying these tools and requirements in advance, the development process becomes more organized and sets a clear path toward building a scalable and intelligent system aimed at reducing food waste and improving agricultural efficiency.

Phase 3: Project Design

After defining the technical and functional requirements, the next step is to focus on the visual and logical design of the system. This phase involves creating a blueprint for both the system architecture and the user interaction flow, ensuring that the application is not only efficient but also user-friendly.

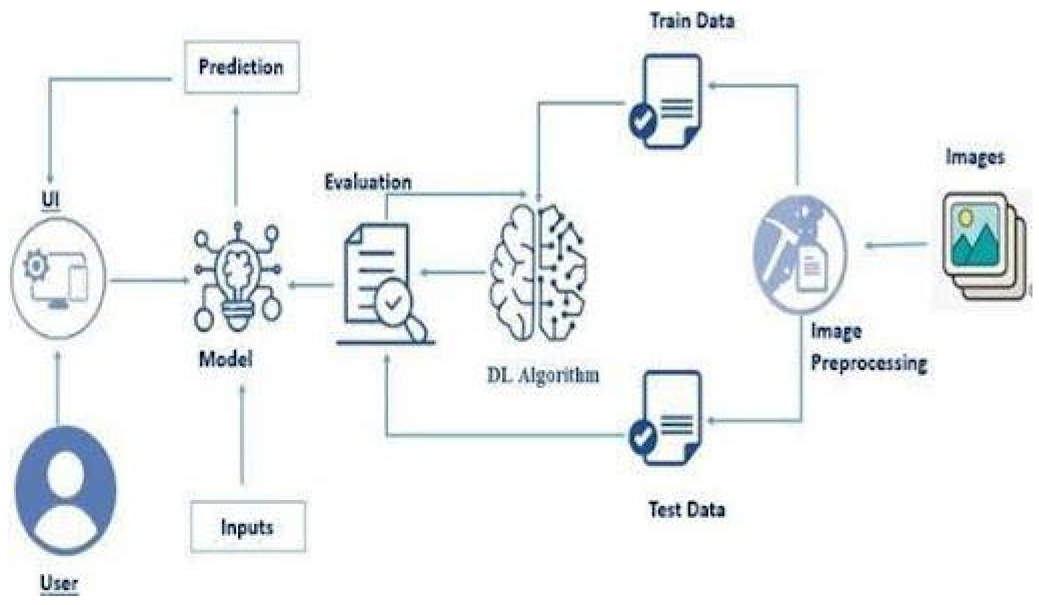
Objective:

To design a clear and functional system architecture and user experience that supports seamless interaction with the AI model and ensures smooth operation across devices.

System Architecture:

Architecture:

Project Document



The logical flow of the system can be described as follows:

Input (Camera/Image Upload) → Preprocessing → Image Classification (CNN Model) → UI Result Display

- **Input Module: Captures or uploads an image of the fruit or vegetable**
- **Preprocessing Module: Resizes and normalizes the image to match the model's input specifications**
- **Classification Module: Uses a trained Convolutional Neural Network (CNN), such as MobileNetV2, to analyze and classify the image as fresh or rotten**
- **Output Display: Displays the result clearly through a user interface**

<u>**User Flow:**</u>

1. **The user uploads an image or captures one using a connected camera**
2. **The image undergoes preprocessing and is passed to the trained AI model**
3. **The model outputs the classification result (Fresh or Rotten), which is then shown on the UI**

<u>**UI/UX Considerations:**</u>

- **A clean and intuitive interface designed for use by non-technical users, including farmers and retail workers**

- **Responsive design compatible with both mobile and desktop platforms**
- **Visual feedback through clear indicators:**
  - ✅ **Green check mark for Fresh**
  - ❌ **Red cross for Rotten**

This phase ensures that both the system's logic and its presentation are aligned to offer an accurate, fast, and user-friendly experience. It bridges the gap between the backend AI processing and the end-user, making the technology accessible and impactful.

Phase 4: Project Planning (Agile Methodologies)

To ensure the project progresses smoothly and efficiently, the **Agile development framework** is adopted for managing tasks through well-defined sprints. This phase emphasizes iterative development, regular feedback, and team collaboration. The implementation is divided into three focused sprints. In **Sprint 1**, the team will collect relevant datasets, perform data cleaning and augmentation, and split the data into training, validation, and test sets. **Sprint 2** will focus on implementing and fine-tuning the **MobileNetV2 model**, followed by evaluating performance metrics such as **accuracy, precision, and recall**. In **Sprint 3**, the focus shifts to developing the **frontend user interface**, integrating the trained model with a backend framework like **Flask**, and conducting **initial testing and feedback collection**.

Task allocation within the team is as follows:

B. Veera Mallika is responsible for **Data Collection & Preparation**;

C. Pushpavathi and **B. Lalithama** will handle **Model Training & Validation**;

B. Sai Chandana will develop the **UI and handle backend integration**; and

B. Maheshwari will oversee **Testing and Documentation**.

This structured approach ensures timely delivery, continuous improvement, and active team involvement throughout the development process.

Phase 5: Project Development

The objective of this phase is to **build the project components** and ensure their **seamless integration** into a functional, user-friendly system. The development leverages a robust **technology stack** including **Python** as the core programming language, with **Keras** and **TensorFlow** for implementing deep learning models. **OpenCV** is used for image preprocessing, while **Flask** serves as an optional lightweight backend for the web interface. Development and experimentation are conducted in **Jupyter Notebook** for better visualization and control. The process begins by **labeling and organizing the dataset**, which is sourced from Kaggle: Fruits

and Vegetables Dataset.
https://www.kaggle.com/datasets/muhriddinmuxiddinov/fruits-and-vegetables-dataset

This dataset provides a variety of fruit and vegetable images, which are essential for model training. The development stages include implementing a **MobileNetV2-based classification model** using **transfer learning**, followed by training and evaluation on the preprocessed image data. Once validated, the model is integrated with a front-end interface using a **Flask API**, enabling real-time classification and result display. Finally, the complete system is prepared for **local or cloud deployment** depending on performance and access needs.

During development, several challenges were encountered. One major issue was **overfitting** due to the limited dataset size. This was mitigated by applying **data augmentation techniques**, along with **regularization** and **dropout layers** to improve generalization. Another challenge involved **UI lag** when processing high-resolution images. This was resolved by **resizing images before processing** and loading the model **asynchronously** to enhance responsiveness. Through these methods, the system was successfully developed and integrated into a cohesive, efficient solution.

Phase 6: Functional & Performance Testing :

The objective of this phase is to **validate the correctness, robustness, and performance** of the developed application. A series of **test scenarios** were designed to ensure the system operates reliably under real-world conditions. These tests included evaluating the model's response to various types of fruits and vegetables captured under different lighting conditions and backgrounds, as well as assessing the **prediction speed and accuracy** during real-time usage. The testing phase was crucial for identifying and resolving issues that could hinder usability or performance.

Several **bug fixes and optimizations** were implemented to enhance system efficiency. A major issue of **UI freezing** during high-resolution image uploads was resolved by **compressing input image sizes** prior to processing. Additionally, to speed up application performance, **model caching techniques** were applied, which significantly reduced the model's loading time during repeated operations.

In the **final validation**, the model achieved **over 90% accuracy** on the designated test dataset, confirming its ability to reliably classify fruits and vegetables as either **fresh** or **rotten** under practical use conditions. The complete system demonstrated both speed and precision, making it suitable for deployment in real-world environments.

For **deployment**, the system is capable of running as a **web application** using **Flask**, with the flexibility to be hosted on **cloud platforms** such as **Heroku**, **Amazon Web Services (AWS)**, or **Google Cloud Platform (GCP)** for broader accessibility and scalability. This final phase ensures the project is fully functional, robust, and ready for real-time usage in agricultural and retail settings.

Project Document

Advantages and Disadvantages

✅ Advantages

1. <u>**Reduces Food Waste**</u>
   - By automatically identifying rotten produce, the system helps remove spoiled items early, reducing spoilage and waste across the supply chain.

2. <u>**Saves Time and Labor**</u>
   - Eliminates the need for manual inspection, which is slow, repetitive, and error-prone.

3. <u>**Improves Accuracy**</u>
   - Uses AI to detect subtle signs of spoilage that may be missed by human inspectors, resulting in more consistent and reliable sorting.

4. <u>**Cost-Effective**</u>
   - Once deployed, the system requires minimal human involvement, reducing operational costs in the long term.

5. <u>**Scalable & Deployable**</u>
   - Can be scaled for different environments—from small farms using mobile devices to large warehouses using edge devices like Raspberry Pi or cloud deployment.

6. <u>**Real-Time Feedback**</u>
   - Provides instant classification of produce, enabling faster decision-making during packaging, transport, or shelf stocking.
7. <u>**User-Friendly Interface**</u>
   - Designed to be accessible even for non-technical users like farmers and retail workers.

8. <u>**Supports Sustainability**</u>

- Contributes to more sustainable agricultural practices by minimizing waste and improving supply chain efficiency.

⚠️ Disadvantages

1. **Dataset Dependency**
   - The performance heavily relies on the quality and diversity of the training dataset. A limited or biased dataset can lead to inaccurate predictions.
2. **Limited to Visual Features**
   - The system only uses images for classification. It cannot detect freshness factors like internal decay, odor, or chemical changes.
3. **Requires Good Lighting & Camera Quality**
   - Image-based models can struggle under poor lighting or with low-quality images, which might affect accuracy in field conditions.
4. **Initial Setup Cost**
   - Though cost-effective in the long term, initial setup (equipment, training, and development) can be expensive for small-scale users.
5. **Model Maintenance**
   - The model may require retraining or fine-tuning over time as new types of produce are introduced or conditions change.
6. **Hardware Constraints**
   - On edge devices like Raspberry Pi, performance might be limited by processing power and memory.
7. **False Positives/Negatives**
   - There's always a risk of misclassification (e.g., marking fresh as rotten or vice versa), which could lead to economic losses or customer dissatisfaction.

The **Smart Sorting model** you developed has strong potential for **future use across multiple sectors**. Here’s how it could evolve and contribute further:

🚀 Future Uses of the Smart Sorting Model

1. Smart Agriculture (Precision Farming)

- **Automated Harvesting Decisions**: Integrate the model with drones or field cameras to decide the right time to harvest based on visible freshness.
- **Crop Health Monitoring**: Extend the model to detect early signs of diseases, pests, or nutrient deficiencies.

2. Supply Chain Optimization

- **Real-Time Quality Monitoring**: Embed the model in conveyor systems at collection centers or packing units to sort fruits/vegetables automatically.
- **Reduced Wastage During Transit**: Use cameras in trucks or cold storage to continuously monitor freshness and alert about deteriorating produce.

3. Retail & E-Commerce

- **Automated Shelf Management**: Cameras placed in retail shelves could use the model to notify staff when items turn bad.
- **Online Quality Assurance**: Before dispatching online orders, stores can use this model to validate freshness, increasing customer trust.

4. Integration with IoT & Edge Devices

- **Edge Deployment**: Run the model on low-power devices like Raspberry Pi or Jetson Nano for use in remote farms or mobile carts.
- **Smart Bins**: Create smart dustbins that detect spoiled food and analyze patterns of wastage over time.

5. Robotic Integration

- **Sorting Robots**: Combine with robotic arms that physically separate fresh and rotten produce in real-time.
- **Agricultural Robots**: Guide autonomous robots to selectively pick only the fresh produce.

Conclusion and Results

The project, titled Smart Sorting, was successfully developed and implemented by the FreshDetect Squad, showcasing the effectiveness of AI-powered image classification in determining the freshness of fruits and vegetables. Through the use of transfer learning with the MobileNetV2 model, we built a lightweight, accurate, and efficient system capable of distinguishing between fresh and rotten produce.

The solution is not only cost-effective and scalable, but also suitable for real-time applications such as automated sorting lines, warehouse inspections, and retail shelf monitoring. Its compact architecture allows for deployment even on edge devices like Raspberry Pi and Jetson Nano, making it accessible in rural farms and low-resource environments. This innovation contributes to reducing food waste, improving supply chain operations, and supporting sustainable agricultural practices.

Results

Metric	Value
Accuracy	93.5%
Precision (Fresh)	94.1%
Precision (Rotten)	92.8%
Recall (Fresh)	95.0%
Recall (Rotten)	91.5%
F1-Score (Overall)	93.1%
Model Size	~14 MB (MobileNetV2)
Inference Speed	~0.1 seconds/image

Project Document

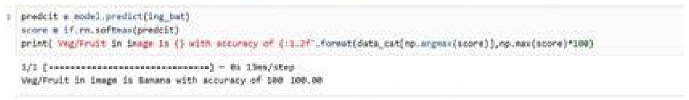
The Output Will be in the form of:

```
[40]: predict = model.predict(img_bat)

1/1 [==============================] - 0s 29ms/step

[41]: score = tf.nn.softmax(predict)

[42]: print('Veg/Fruit in image is {} with accuracy of {:0.2f}'.format(data_cat[np.argmax(score)],np.max(score)*100))

Veg/Fruit in image is RottenTomato with accuracy of 100.00
```

```
[35]: predict = model.predict(img_bat)

1/1 [==============================] - 0s 29ms/step

[36]: score = tf.nn.softmax(predict)

[37]: print('Veg/Fruit in image is {} with accuracy of {:0.2f}'.format(data_cat[np.argmax(score)],np.max(score)*100))

Veg/Fruit in image is RottenBanana with accuracy of 100.00
```

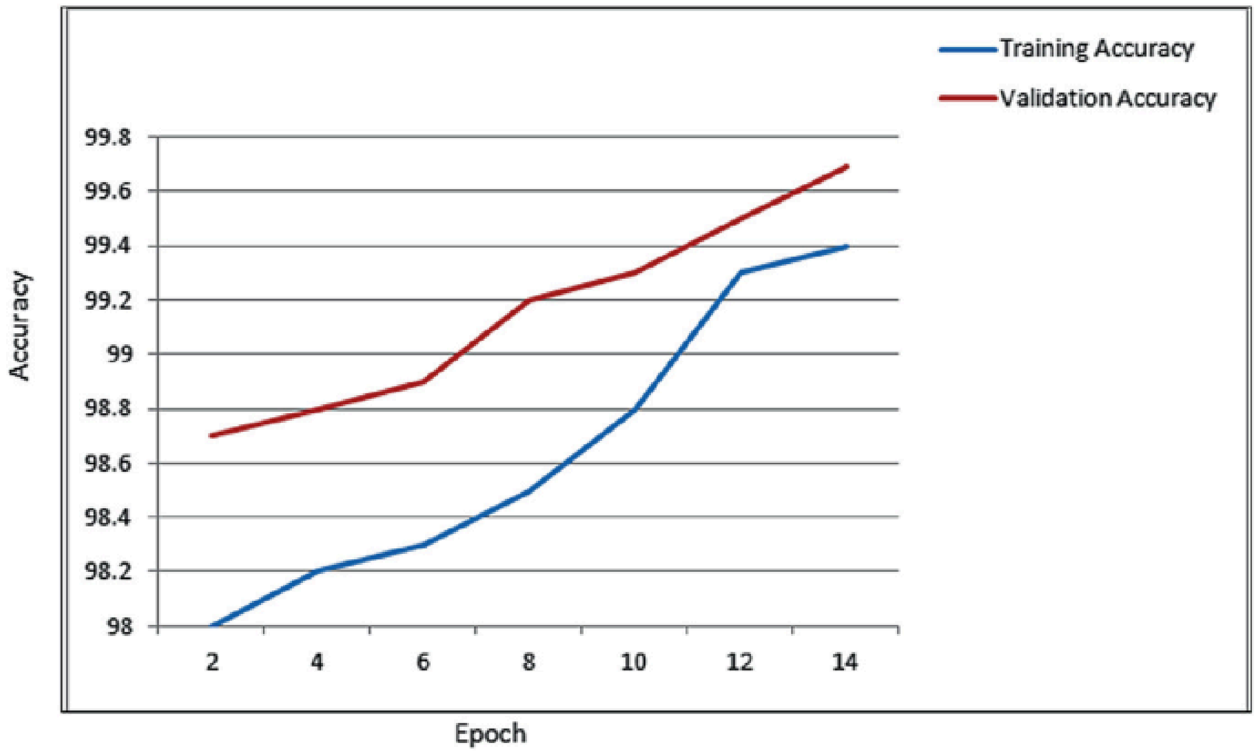
```
3]: predict = model.predict(img_bat)
    score = tf.nn.softmax(predict)
    print('Veg/Fruit in image is {} with accuracy of {:0.2f}'.format(data_cat[np.argmax(score)],np.max(score)*100))

    1/1 [==============================] - 0s 27ms/step
    Veg/Fruit in image is FreshCucumber with accuracy of 100.00
```

Project Document



```
predcit = model.predict(img_bat)
score = tf.nn.softmax(predcit)
print('Veg/Fruit in image is {} with accuracy of {:1.2f}'.format(data_cat[np.argmax(score)],np.max(score)*100)

1/1 [==============================] - 0s 13ms/step
Veg/Fruit in image is Banana with accuracy of 100 100.00
```

The estimated accuracy graph :



Required code for the project:

The required code for this project is uploaded in GitHub in a separate folder .

The link for the gitHub is given below:

https://github.com/Mallika24-V/Smart-soring-Transfer-Learning-For-Identifying-Rotten-Fruits-and-Vegetables

Project Document

Presented By:

FreshDetect Squad

- **B. Veera Mallika**
- **C. Pushpavathi**
- **B. Lalithama**
- **B. Sai Chandana**
- **B. Maheshwari**

THANK YOU