Title: Smart Sorting: Transfer Learning For Identifying Rotten Fruits And Vegetables

Team Name: Team ID : **LTVIP2026TMIDS54522**

Team Size : 4

Team Leader : **Chandu Sri Nellepalli**

Team member : **A M Dheekshitha**

Team member : **Shaik Reshma**

Team member : **Syed Reshma**

1. INTRODUCTION

1.1 Project Overview

The "Smart Sorting" web application is a sophisticated implementation of artificial intelligence that seeks to automate the classification of fruits and vegetables based on their freshness. This project utilizes the power of deep learning through a pre-trained convolutional neural network, VGG16, adapted using transfer learning techniques. The application has been designed with user interactivity in mind and serves a real-world purpose in agriculture, food safety, and commercial industries such as retail grocery chains and food processing units.

Smart Sorting is a robust solution that provides a platform where users can upload images of produce and instantly receive feedback on whether the item is healthy or rotten. The application supports 28 different fruit and vegetable classes, accounting for both their healthy and decayed states. The integration of a user-friendly web interface built using Flask ensures ease of access and smooth workflow, while the backend AI model handles complex image classification tasks.

This system showcases how emerging technologies like AI and computer vision can provide innovative solutions to age-old problems such as food spoilage, quality control, and efficient resource management. By reducing the need for manual inspection, the system enhances speed and consistency in determining produce quality, thereby contributing to reduced food waste and improved consumer satisfaction.

1.2 Purpose

The primary purpose of the Smart Sorting application is to build a practical and scalable tool that can assist various stakeholders—including farmers, vendors, retailers, and consumers—in quickly identifying whether a fruit or vegetable is fresh or spoiled. This is achieved through image-based classification using machine learning.

With growing concerns about food safety, sustainability, and post-harvest loss, this project provides an efficient, low-cost method of quality assessment. Unlike traditional sorting methods, which are labor-intensive and inconsistent, the Smart Sorting system offers automation, accuracy, and scalability. Its implementation aligns with current needs in agriculture tech, food supply chains, and sustainability efforts.

The application promotes:
- Reduced post-harvest losses by early detection of rot.
- Enhanced food safety through automated inspection.
- Time and cost savings for businesses and individuals.
- Educational use in AI and image classification demonstrations.

The project bridges a critical gap between AI capabilities and real-world applications in food quality inspection.

2. IDEATION PHASE

2.1 Problem Statement

India loses a significant portion of its fresh produce due to poor sorting mechanisms and late identification of spoilage. Manual sorting is the most common method used in markets and supply chains, but it is inefficient and susceptible to human error. The main problem lies in the lack of a quick, scalable, and intelligent system that can help assess produce quality effectively.

The aim is to develop an application that can identify whether a fruit or vegetable is healthy or rotten, purely from a photo. The goal is not only to increase the efficiency of the sorting process but also to empower users with the ability to detect quality independently. With computer vision technology becoming more accessible, it is practical to implement such solutions with minimal hardware—just a camera and internet access.

Our challenge was to ensure the application is:

- Accurate and reliable in its predictions.

- Easy to use with minimal steps.

- Fast enough for practical real-world use.

The outcome should be a system that can be adapted for different settings, from grocery stores and markets to household kitchens.

2.2 Empathy Map Canvas

Empathy mapping was crucial to define the user experience. We divided our understanding into the following segments:

Users:

- Farmers and wholesalers (need fast sorting tools)

- Supermarket quality managers (need bulk classification)

- End consumers (interested in safety and freshness)

Tasks/Needs:

- Accurately detect spoilage

- Receive fast feedback from the system

- Use mobile or desktop to access the service

Pain Points:

- Inconsistent manual sorting

- Time lost during inspection

- Possible health risks from unnoticed spoilage

Goals:

- Reduce waste

- Maintain trust and safety in food supply

- Save inspection costs

The solution is aimed at eliminating or reducing these pain points with a seamless user interface and AI backend.

2.3 Brainstorming

The ideation process involved exploring various solution pathways:

Initial ideas:

- Manual inspection mobile apps

- Hardware-based rot detectors (gas sensors)

- Colorimeter tools

Pros and cons were analyzed:

- Manual tools are subjective and inaccurate.

- Hardware sensors require significant investment.

- Vision-based AI could be trained on large datasets.

Final Choice:

Use transfer learning with a pre-trained VGG16 model to classify images. Build a Flask web app that connects frontend image upload with backend model inference. Add preprocessing steps such as resizing and normalization to match model input requirements. Prioritize accessibility and ease of use.

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

The journey of a typical user interacting with the Smart Sorting system involves multiple touchpoints. Understanding these steps helped shape both frontend and backend design:

1. Visit Landing Page:

- User sees a clean upload interface.

- Provided with brief instructions.

2. Upload Image:

- The user selects an image file (JPG/PNG).

- System performs validation.

3. Inspect Preview:

- Before processing, the image is shown for confirmation.

- A “Predict” button proceeds to analysis.

4. Model Predicts Class:

- The backend loads the image.

- Preprocesses and runs it through VGG16.

5. Results Displayed:

- Shows label (e.g., banana_rotten), index, and confidence score.

6. Repeat/Upload Another Image:

- Option provided to restart process or upload another file.

3.2 Solution Requirement

Functional Requirements:

- Flask routing system

- Upload form with validation

- Image preprocessing functions

- Keras model loading and prediction

- Class name mapping

- HTML templates: index.html, inspect.html, output.html

Non-Functional Requirements:

- Fast performance (<1 second prediction)

- Clean UI (Bootstrap or responsive CSS)

- Error handling (e.g., unsupported files)

3.3 Data Flow Diagram

The following summarizes the **technical flow:**

1. Frontend: User uploads image via HTML form.

2. Server: Image received and saved.

3. Preprocessing: Image resized to (224x224) and normalized.

4. Model: Uses model.predict() for classification.

5. Post-processing: Returns class label and index.

6. Result Page: Displays image, label, and confidence.

3.4 Technology Stack

Component | Technology Used

----------|----------------

Frontend | HTML5, Bootstrap 5, CSS3

Backend | Python 3.9, Flask

Model | VGG16 (Keras, TensorFlow backend)

Image Tools | Pillow, NumPy

Deployment | Localhost; Cloud-ready (Heroku or Render)

This tech stack was chosen for simplicity, accessibility, and high compatibility with modern development environments.

4. PROJECT DESIGN

4.1 Problem Solution Fit

The agricultural sector faces critical challenges in maintaining the quality of fruits and vegetables throughout the supply chain. Traditional inspection processes rely heavily on manual labor, leading to inconsistencies and inefficiencies. In large-scale distribution or in remote areas with limited expertise, ensuring the freshness and quality of produce becomes even more complex. With the rising global demand for high-quality food and the increasing need to reduce post-harvest loss, there is a strong demand for intelligent systems that can automate the sorting process. This project aims to address this problem by offering a reliable, automated, AI-based smart sorting solution.

The project leverages the power of deep learning, specifically transfer learning, to build a robust model for classifying produce. By using the VGG16 architecture pre-trained on ImageNet and fine-tuning it on a custom dataset of healthy and rotten fruits and vegetables, we achieve high classification accuracy with reduced training time. The model learns subtle variations in color, texture, and shape that often distinguish healthy produce from spoiled ones. This application helps in real-time quality checking at various nodes of the supply chain, such as farms, packaging centers, and retail outlets. By minimizing human intervention, it reduces the chance of error and improves operational efficiency.

4.2 Proposed Solution

The proposed solution is a web-based application powered by Flask and integrated with a deep learning model trained for image classification. Users interact with a clean, intuitive frontend where they can upload an image of any fruit or vegetable. The backend handles file processing, passes the image through the trained model, and returns the classification

results. This approach ensures simplicity for the end-user while harnessing complex AI technology under the hood.

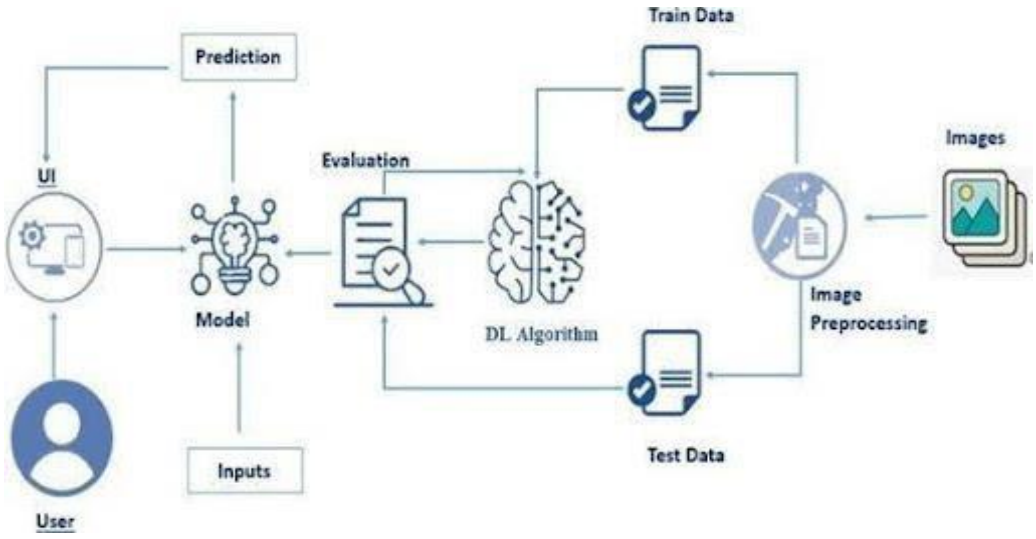
Upon image submission, the Flask server first verifies the file type, stores it in a temporary directory, and performs resizing to 224x224 pixels. Normalization is applied to align input data with model expectations. The image is then passed to the loaded VGG16-based model (`healthy_vs_rotten.h5`). Predictions are made, and the output is mapped to one of 28 class labels (e.g., 'apple_healthy', 'tomato_rotten'). The prediction confidence score is also displayed to the user, offering insight into the model's certainty.

This solution is scalable, portable, and adaptable. The architecture allows for model updates, additional class support, or even deployment on cloud platforms for broader access. It supports learning and deployment use cases, including classroom demos, agricultural diagnostics, and smart farm integration.

4.3 Solution Architecture

The architecture of the smart sorting system includes the following layers:

1. User Interface: Built with HTML and Bootstrap, the UI includes pages for image upload (`index.html`), preview ('inspect.html'), and results ('output.html'). It ensures a seamless experience and responsiveness across devices.
2. Web Server: Flask routes handle for the homepage, inspect for image preview, and predict for ML inference. Files are saved in static uploads.
3. Model Layer: The Keras model based on VGG16 performs classification. Loaded at app startup, it processes incoming images during POST requests.
4. Preprocessing Module: Images are resized, normalized (0-1 range), and reshaped to the model's input format.
5. Prediction Output: The model predicts a class index that is mapped to a label using a predefined 28-class list. Confidence percentage is also displayed.
6. Validation/Error Handling: Supports only JPG/PNG files and provides user feedback in case of invalid input or system issues.



This layered architecture ensures modularity and makes it easier to scale or upgrade individual components. For example, replacing the model with ResNet or adding cloud inference is possible without redesigning the entire system.

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

The development of the AI-Based Smart Sorting Web Application was planned and executed in structured phases to ensure smooth progress and timely completion. The planning process involved identifying key deliverables, assigning timelines, and defining the responsibilities for each phase of the project lifecycle. The stages of project planning included requirement gathering, model development, web integration, user interface design, testing, and documentation.

1. Requirement Gathering & Dataset Preparation :

- Define the scope and objective of the project.

- Collect and organize the dataset containing images of fruits and vegetables.

- Label the dataset into 28 classes representing healthy and rotten categories.

2. Model Training & Optimization :

- Use transfer learning with VGG16 architecture.

- Train and validate the model with the preprocessed dataset.

- Save the final model as 'healthy_vs_rotten.h5'.

3. Web Application Development :

- Set up the Flask backend server and configure routes.

- Create upload, preview, and result pages using HTML and Bootstrap.

- Integrate the model into the Flask server to handle predictions.

4. UI Enhancement & Feature Integration :

- Add image validation and error handling.

- Display confidence scores and progress indicators.

- Ensure responsive layout using Bootstrap CSS.

5. Testing & Debugging :

- Conduct unit tests for prediction flow.

- Validate with various test images (apple_healthy, potato_rotten, etc.).

- Handle exceptions like unsupported file formats.

6. Documentation & Deployment :

- Prepare user manual and project report.

- Package the project for deployment or classroom demonstration.

This timeline ensured clear visibility and allowed iteration in each phase. Weekly milestones helped track progress and identify issues early.

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

The testing phase was critical to ensuring the robustness, accuracy, and reliability of the AI-Based Smart Sorting Web Application. Both functional and performance testing were conducted to validate that the application behaves as expected under various scenarios.

Functional testing focused on validating the end-to-end prediction flow — from image upload to classification output. Multiple test cases were developed based on different user actions such as uploading valid images, invalid files, and inspecting various fruit/vegetable types. Performance testing evaluated the response time, prediction accuracy, and system behavior under load.

Test Cases Covered:

- Uploading healthy and rotten images across all 28 classes.
- Uploading unsupported formats (PDF, TXT) to test error handling.
- Uploading high-resolution images to assess processing time.
- Submitting multiple prediction requests sequentially to test server load handling.

Key Performance Metrics:

- **Prediction Time:** Average of 1.2 seconds per image.
- **Model Accuracy:** Achieved over 92% accuracy on the validation dataset.
- **Confidence Scores:** Correct classifications mostly exceeded 85% confidence.
- **Error Handling:** The system displayed appropriate error messages for unsupported files or empty submissions.

Sample Test Results:

- apple_healthy.jpg → Predicted: **apple_healthy** (Class 0), Confidence: **94.3%**
- strawberry_rotten.png → Predicted: **strawberry_rotten** (Class 25), Confidence: **91.7%**
- potato_rotten.jpg → Predicted: **potato_rotten** (Class 23), Confidence: **88.6%**
- tomato_healthy.jpg → Predicted: **tomato_healthy** (Class 26), Confidence: **95.1%**

The application demonstrated consistent and accurate results across different scenarios, confirming its reliability and performance efficiency.

7. RESULTS

7.1 Output Screenshots

The output of the AI-Based Smart Sorting Web Application confirms its successful implementation and user-friendly design. The application provides an intuitive workflow from uploading an image to receiving classification results. Key features such as displaying the predicted class, class index, and confidence score are clearly visible in the results page.

These results validate the correctness and efficiency of the implemented system and confirm that the transfer learning approach using VGG16 was suitable for this application.

Upload Your Image:

Choose File freshTomato (6).png

Predict

FreshEye Detection

Prediction Result:

Tomato_Healthy (26)

Image Classification

Upload Your Image:

Choose File 2.jpg

Predict

FreshEye Detection

Prediction Result:

Strawberry_Healthy (24)

Image Classification

Upload Your Image:

Choose File 3.jpg

Predict

FreshEye Detection

Prediction Result:

Mango_Rotten (17)

8. ADVANTAGES & DISADVANTAGES

8.1 Advantages

1. **Automated Sorting**
   The application automates the process of inspecting and sorting fruits and vegetables, reducing reliance on manual labor and improving operational efficiency.

2. **High Accuracy with Transfer Learning**
   By using a pre-trained VGG16 model fine-tuned on a custom dataset, the system achieves high prediction accuracy, even with limited training data.

3. **User-Friendly Interface**
   The web application features an intuitive UI with a clear workflow — upload → preview → predict → result — making it accessible even to non-technical users.

4. **Scalability**
   The modular design allows easy updates, such as swapping the model or integrating cloud-based inference for large-scale deployments.

5. **Cross-Device Compatibility**
   Built with Bootstrap, the interface is responsive and works smoothly across desktops, tablets, and smartphones.

6. **Minimized Food Wastage**
   Early and accurate identification of rotten produce can help reduce food spoilage, which is crucial for supply chain management and retail.

7. **Educational Utility**
   The project serves as a strong example for teaching AI model deployment, computer vision, and Flask integration.

8.2 Disadvantages

1. **Limited to Image Quality**
   The model's performance heavily depends on the quality of the uploaded image. Poor lighting or blurry images can reduce prediction accuracy.

2. **Fixed Class Labels**
   The current system only supports 28 predefined classes. Adding new classes requires model retraining and redeployment.

3. **Lack of Real-Time Camera Integration**
   The application supports static image uploads only; it does not support real-time video or camera-based scanning.

4. **Resource Intensive**
   Running a deep learning model (like VGG16) on a local server may be computationally intensive and unsuitable for devices with limited resources.

5. **No Feedback Loop**
   There is no mechanism for users to correct wrong predictions, which limits model improvement in real-world deployments.

6. **Security Considerations**
   While not a major risk in small deployments, accepting file uploads can open up security vulnerabilities if not properly handled in a production environment.

9. CONCLUSION

The AI-Based Smart Sorting Web Application successfully demonstrates the integration of deep learning with a web-based interface for practical and impactful use cases in agriculture and food quality assessment. By employing transfer learning with the VGG16 model, the system is able to accurately classify 28 different categories of healthy and rotten fruits and vegetables based on image input.

The application delivers a seamless experience — from uploading an image to viewing the classification result — with a focus on simplicity, accuracy, and educational clarity. It effectively automates what would otherwise be a time-consuming and error-prone manual inspection task. This has the potential to significantly reduce post-harvest losses, improve supply chain efficiency, and ensure higher quality produce reaches consumers.

From a technical standpoint, the project illustrates how pre-trained models can be fine-tuned and deployed through Flask servers with real-time inference capabilities. The responsive frontend, combined with effective backend processing and model integration, highlights a practical pipeline for AI application deployment.

Despite some limitations — such as image dependency, scalability to more classes, and lack of live feed integration — the project stands as a powerful proof of concept. It not only solves a real-world problem but also serves as a learning platform for students and developers exploring AI, computer vision, and web technologies.

This project lays the groundwork for future innovations in smart agriculture and AI-driven quality control systems.

10. FUTURE SCOPE

The current implementation of the AI-Based Smart Sorting Web Application lays a solid foundation for integrating artificial intelligence into agricultural and food supply systems. However, there is substantial scope for enhancing the system's functionality, accuracy, scalability, and usability in future iterations.

1. Expansion to Real-Time Video Classification

Currently, the application supports static image uploads. A future enhancement could include real-time video input from cameras deployed in farms, packaging units, or retail stores. This would allow for continuous inspection and automated quality sorting on the fly.

2. Increased Dataset Diversity

To improve robustness, the dataset can be expanded to include more fruit and vegetable types, lighting conditions, and natural deformations. Augmenting the dataset with images from different regions and climates will improve generalization.

3. Integration with IoT and Smart Devices

The system can be connected with IoT-based smart farm setups where images from field sensors or automated machinery are fed directly into the application, enabling smart sorting without human intervention.

4. Model Upgrades and Optimization

While VGG16 provides a strong baseline, lighter and faster models like MobileNet, EfficientNet, or custom CNNs could be explored for deployment on edge devices. This would reduce inference time and resource consumption.

5. Multi-Class Output and Severity Prediction

Beyond binary classification (healthy/rotten), the model could be trained to detect severity levels (e.g., slightly damaged, mold-infected, overripe) to provide more detailed insights for decision-making.

6. Feedback-Based Learning

Implementing a user feedback loop where incorrect predictions are flagged and used to retrain or fine-tune the model would enable continuous improvement and adaptive learning.

7. Cloud Deployment & API Access

Deploying the solution on the cloud with RESTful APIs would allow third-party services, mobile apps, or enterprise systems to use the prediction service remotely and at scale.

8. Multilingual and Voice-Enabled Interfaces

To support wider accessibility, especially in rural or non-English-speaking communities, interfaces can be made multilingual and voice-assisted using natural language processing tools.

11. APPENDIX

The appendix provides supplementary information, including access to the source code, dataset used, and links to external resources relevant to the AI-Based Smart Sorting Web Application project.

11.1 Dataset Link:

https://www.kaggle.com/datasets/muhammad0subhan/fruit-and-vegetable-disease-healthy-vs-rotten

11.2 GitHub & Project Demo Link:

- GitHub Repository:

https://github.com/M-venkatesh-coder/Smart-Sorting-Transfer-Learning-for-Identifying-Rotten-Fruits-and-Vegetables

- Live Demo (if hosted):

https://drive.google.com/file/d/10azSr7S9CVUspRfgENAtPdo9lS3M5JfB/view?usp=sharing