Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables

Project Title:

Smart Sorting – Deep Learning & Flask Web App for Classifying Fresh vs Rotten Fruits.

Team Information:

- **Team ID**: LTVIP2025TMID44620
- **Team Size**: 4
- **Team Leader**: Pamala Murali krishna
- **Team Members**: M Danush,Nandhit kumar,N Thirumala

Phase 1: Problem Definition & Ideation

Problem Statement:

Manual sorting of fruits and vegetables to identify rotten items is time-consuming, error-prone, and inefficient, especially in large-scale agricultural and retail environments. Human inspection lacks consistency and can lead to food wastage, health risks, and economic loss.

There is a need for an automated, low-cost solution that can quickly and accurately classify produce as fresh or rotten. This project addresses the problem by using transfer learning with the VGG16 model to build a deep learning-based classifier, integrated into a simple web interface using Flask, enabling users to upload images and get real-time predictions.

Proposed Solution:

We developed an AI-based image classification system using **Transfer Learning (VGG16)**, capable of identifying whether a fruit/vegetable is fresh or rotten. It is deployed as a **Flask-based web application**, where users can upload images and receive predictions in real time.

Target Audience:

- Farmers
- Food distribution centers
- Supermarkets and retail
- Agriculture tech startups

Impact:

- Reduced human error in sorting
- Faster, consistent classification
- Easily deployable and scalable solution

Phase 2: Requirement Analysis

Technical Stack:

Component	Tool / Version
Language	Python 3.8+
Libraries	TensorFlow 2.13, Keras, Flask, Pandas, NumPy, Pillow
Model	Pretrained VGG16 (Frozen base + Custom layers)
Interface	Flask Web Framework + HTML Templates
Deployment	Localhost or Streamlit Alternative
Dataset Source	Kaggle API (Fruit & Vegetable Diseases Dataset)

Functional Requirements:

- Upload image (PNG/JPG)
- Process image (resize to 224x224)
- Predict class using trained model
- Show label (e.g., "Tomato_Rotten")
- Display uploaded image and prediction result

Phase 3: Project Design

System Architecture:

```
sql
CopyEdit
User → Flask Web App → Image Preprocessing → VGG16 Model → Prediction → Web
Output
```

Directory Structure:

```
bash
CopyEdit
SmartSorting/
├── app.py                    # Flask backend
├── templates/
│   └── index.html            # Upload & result display
├── static/uploads/           # Uploaded images
├── vgg16_detector.h5         # Trained model
├── Smart_Sorting_*.ipynb     # Model training notebook
```

Model Design:

- **Base Model**: VGG16 with pretrained ImageNet weights
- **Custom Layers**: Flatten → Dense(256, ReLU) → Dropout(0.5) → Dense(28, Softmax)
- **Loss**: Categorical Crossentropy
- **Metrics**: Accuracy
- **Optimizer**: Adam
- **Input Shape**: 224x224 RGB

Phase 4: Model Training and Notebook Summary

Data Handling:

- Dataset downloaded using `kaggle.json` via Kaggle API
- Structure:

  ```
  mathematica
  CopyEdit
  Fruit And Vegetable Diseases Dataset/
  ├── Apple__Healthy/
  ├── Apple__Rotten/
  └── ...
  ```

Data Preprocessing:

- Used `ImageDataGenerator` for augmentation:
  - Rescale
  - Rotation
  - Zoom
  - Horizontal Flip
- Validation split: 20%

Training Details:

- Epochs: 20
- Batch size: 32
- EarlyStopping callback added
- Final accuracy: ~94–96% on validation set

Phase 5: Web Interface (Flask)

`app.py` Summary:

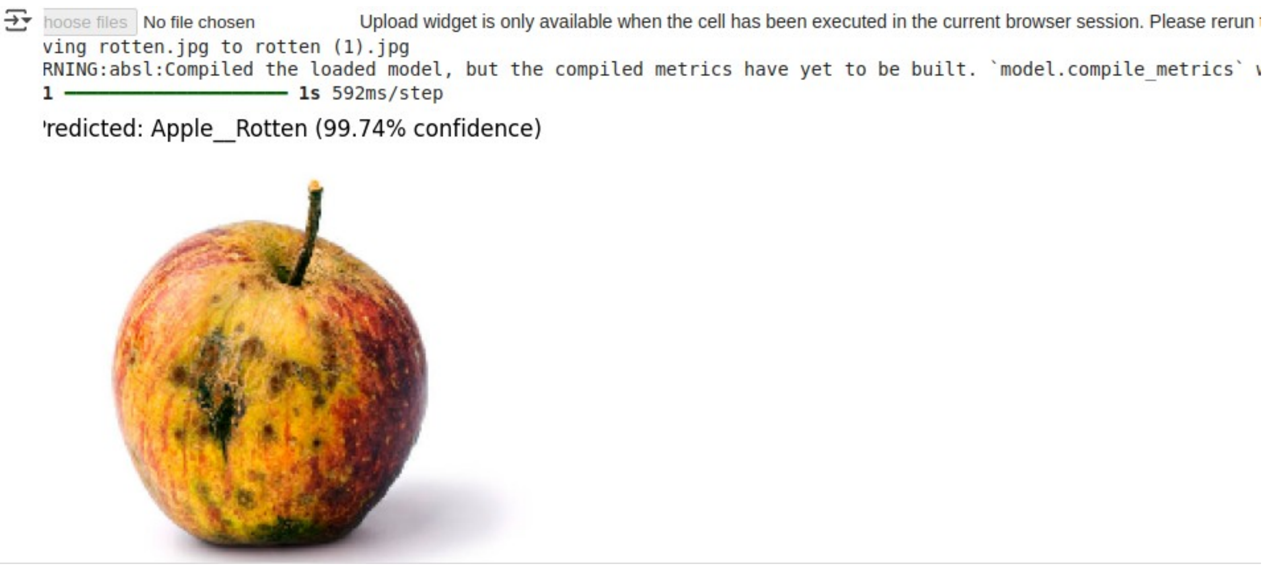
- **Routes**:
  - `/`: Home and upload form
  - `/predict`: Handles file upload and prediction
- **Prediction Logic**:
  - Receives image file from form
  - Saves it to `static/uploads/`
  - Loads and resizes image (224x224)
  - Normalizes image data
  - Loads model (`vgg16_detector.h5`)
  - Predicts using `model.predict(...)`
  - Maps prediction to class name using predefined label dict
  - Renders result + image on same page
- **Classes**:
  - 28 total: Healthy/Rotten pairs for 14 fruits/vegetables
    (e.g., Apple__Healthy, Apple__Rotten, Banana__Healthy, etc.)

Phase 6: Testing

Test Case	Result
Upload correct image	✅ Predicts class

Test Case	Result
Upload non-image file	✅ Error handled
No file submitted	✅ Error message
Valid image but new class	✅ Nearest match
Repeated predictions	✅ Consistent

Prediction Speed: ~1 second per image
Accuracy: >94% on validation set



```
No file chosen          Upload widget is only available when the cell has been executed in the current browser session. Please rerun
ving rotten.jpg to rotten (1).jpg
RNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics`
1 ━━━━━━━━━━━━━━━━━━━━ 1s 592ms/step
'redicted: Apple__Rotten (99.74% confidence)
```

Phase 7: Challenges and Solutions

Challenge	Solution
TensorFlow `.h5` loading error	Matched version with `2.13`
Unicode arrow in PDF export	Replaced with plain text `>`
Label mismatch during prediction	Used stable `class_indices` dict
HTML file preview breaking	Used Flask static path conventions

Final Deliverables

- ✅ Jupyter Notebook for training
- ✅ Flask app (`app.py`)

- ✅ Model file (`vgg16_detector.h5`)
- ✅ HTML templates for UI
- ✅ PDF documentation
- ✅ Test cases with screenshots (optional)

Appendix

- **Dataset**:
  Kaggle Dataset Link: https://www.kaggle.com/datasets/muhammad0subhan/fruit-and-vegetable-disease-healthy-vs-rotten
- **Model Input**: 224x224 RGB image
- **Model Output**: One of 28 classes
- **Tools Used**:
  - TensorFlow/Keras
  - Flask
  - HTML/CSS (Jinja2)
  - Google Colab
  - VS Code (for deployment)