Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables

1. Introduction

Project Title:

Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables

Team Members:

- Leela Sesha Sarvani Veelu
- Tulasi Vishalakshi Kundum
- Challa Divya Teja Naga Venkata Prasad
- Pramodh Surya Pradyumna Avala

2. Project Overview

Purpose:

The purpose of the NutriGaze project is to develop an intelligent image classification system that can automatically detect whether a fruit or vegetable is healthy or rotten using deep learning techniques. The system aims to reduce food waste, improve food quality monitoring, and support consumers, retailers, and farmers in making better decisions about produce quality.

By leveraging transfer learning with a pre-trained VGG16 Convolutional Neural Network (CNN) and integrating it into a user-friendly Flask web application, the project provides a simple platform where users can upload an image and instantly receive classification results along with a confidence score.

Goals:

- **Accurate Classification**
  Build a robust deep learning model capable of classifying 28 categories of fruits and vegetables into healthy and rotten classes with good accuracy.
- **Efficient Image Preprocessing**
  Implement proper image resizing, normalization, and preprocessing techniques to

ensure compatibility with the VGG16 architecture.

- **Transfer Learning Implementation**
  Use a pre-trained CNN model and fine-tune selected layers to improve performance while reducing training time.
- **User-Friendly Interface**
  Develop a simple Flask-based web application that allows users to:
  - Upload an image
  - View prediction results
  - See confidence scores
- **Performance Evaluation**
  Evaluate the model using metrics such as:
  - Accuracy
  - Loss

Features:

The Healthy vs Rotten Fruits and Vegetables Detection System includes the following key features and functionalities:

1. Multi-Class Image Classification

The system classifies images into **28 different categories** consisting of both healthy and rotten fruits and vegetables. It accurately identifies the condition of produce using a deep learning–based Convolutional Neural Network (CNN).

2. Transfer Learning with VGG16

The project uses a pre-trained **VGG16 model** as the base architecture. Transfer learning is applied to leverage previously learned image features, and the final layers are fine-tuned for domain-specific classification. This improves accuracy while reducing training time.

3. Image Preprocessing and Augmentation

To improve model performance and generalization:

- Images are resized to **224 × 224 pixels**.
- Pixel values are normalized using 1./255.
- Data augmentation techniques such as rotation, zoom, and horizontal flipping are applied during training.

4. Web-Based Image Upload and Prediction

A user-friendly Flask web interface allows users to:

- Upload an image of a fruit or vegetable.
- Automatically preprocess the image.
- Generate a classification result.

5. Model Performance Evaluation

The system evaluates model performance using:

- Training and Validation Accuracy
- Loss values across epochs
- Confusion Matrix
- Classification Report (Precision, Recall, F1-Score)

6. 3-Tier Architecture

The project follows a structured architecture:

- **Presentation Layer:** HTML/CSS frontend
- **Application Layer:** Flask backend
- **Model Layer:** Trained CNN model (.h5 file)

This architecture ensures modularity and easier deployment.

7. Scalability and Future Enhancements

The system can be extended to:

- Include additional fruit and vegetable categories.
- Deploy on cloud platforms.
- Integrate mobile applications for real-time detection.

3. Architecture

Frontend:

Flask templates (**HTML, CSS, JavaScript**) are used to design the user interface and result dashboard.

- HTML is used to structure the web pages (Home, About, Upload Page).
- CSS is used for styling and responsive layout.
- JavaScript is used for basic interactivity and form handling.
- The interface allows users to upload an image and view prediction results with confidence score.

- Training accuracy and loss graphs are displayed for model performance visualization.

Backend:

A **Python Flask application** handles routing, image processing, and ML model predictions.

- Receives uploaded images from the frontend.
- Performs preprocessing.
- Loads the trained CNN model (VGG16 transfer learning).
- Generates prediction and confidence score.
- Returns results dynamically to the frontend.
- Implements input validation and basic error handling.

Database / Model Storage:

- Dataset consists of **28 classes** (Healthy & Rotten fruits and vegetables).
- Approximately **3358 training images** and **1120 validation images** used for model training.
- Images are resized and augmented before training.
- The trained model is stored as a **.h5 file** and loaded in the Flask backend for inference.
- No traditional database is currently used; predictions are generated in real-time.

Future Scope:

- Integration with cloud storage for image management.
- Database integration (MongoDB/MySQL) to store prediction history.
- Deployment on cloud platforms (AWS/GCP/Azure) for scalability.

4. Setup Instructions

Prerequisites:

To successfully set up and run the Healthy vs Rotten Fruits & Vegetables Detection System, the following software tools, and packages are required:

Anaconda Navigator:

- Refer to the link below to download Anaconda Navigator
- Link : https://youtu.be/1ra4zH2G4o0

- Python packages:
    - numpy
    - pandas
    - matplotlib
    - scipy
    - scikit-learn
    - tensorflow
    - flask
    - pillow
    - OpenCV

Installation:

- Clone the repository.
- Install dependencies using pip install -r requirements.txt
- Run the model training script (if required) using python train_model.py
- Start the Flask server with python app.py.

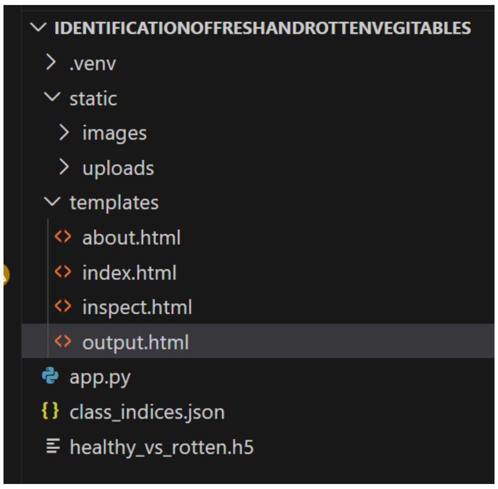
5. Folder Structure

Client:

- templates/ → HTML files (index page, inspect page, about page, result page).
- static/ → images and uploads.

Server:

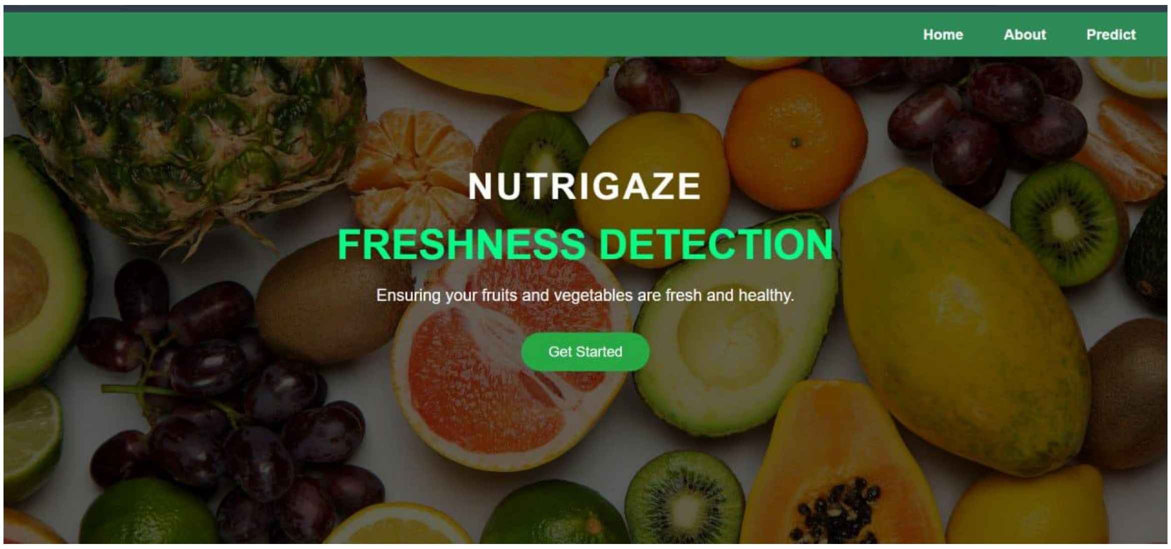
- app.py → Flask application.
- Healthy_vs_rotten.h5 → Saved trained CNN model.
- dataset/ → fresh vs rotten fruits and vegetables image dataset



```
IDENTIFICATIONOFFRESHANDROTTENVEGITABLES
  > .venv
  v static
    > images
    > uploads
  v templates
    about.html
    index.html
    inspect.html
    output.html
  app.py
  class_indices.json
  healthy_vs_rotten.h5
```

6. Running the Application

- Frontend: Runs automatically via Flask templates.
- Backend: Start with python app.py.
- Access at http://127.0.0.1:5000/.



7. API Documentation

Endpoint 1: / (Home Page)

- **Method:** GET
- **Parameters:** None
- **Response:** Renders the homepage with image upload form
- **Output:** User can select and upload a fruit/vegetable image

Endpoint 2: /predict

- **Method:** POST
- **Parameters:**
  - pc_image – Fruit or Vegetable image file (JPG/PNG format)
- **Process:**
  - Image is uploaded and saved in static/uploads/
  - Image resized to **224×224**
  - Preprocessing applied (preprocess_input() for VGG16)
  - Model predicts class among **28 categories**

- Confidence score calculated

- **Response:**
  - Prediction result (e.g., Pomegranate__Healthy)
  - Confidence score (e.g., 94.7%)
  - Uploaded image displayed on result page
- **Output:**
  - Uploaded image name displayed
  - Predicted class displayed
  - Confidence percentage displayed

Endpoint 3: /inspect

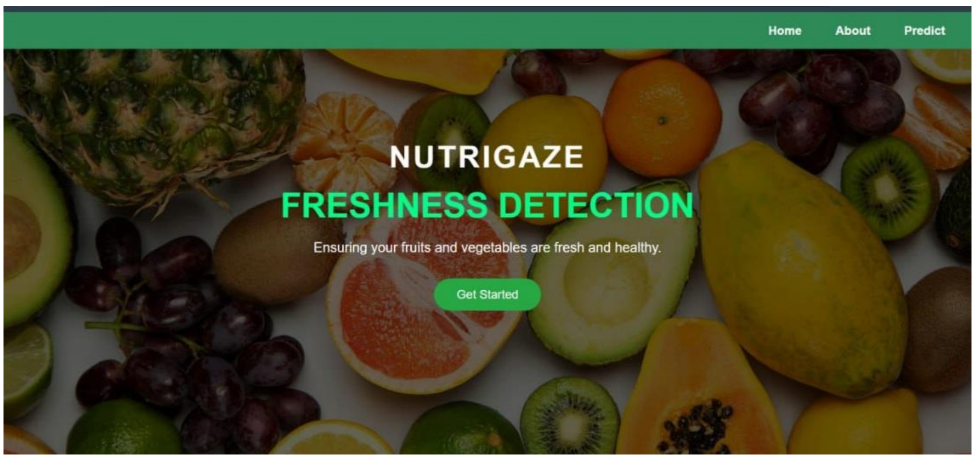
- **Method:** GET
- **Parameters:** None
- **Response:** Loads prediction page interface
- **Output:** Allows user to upload another image

8. Authentication

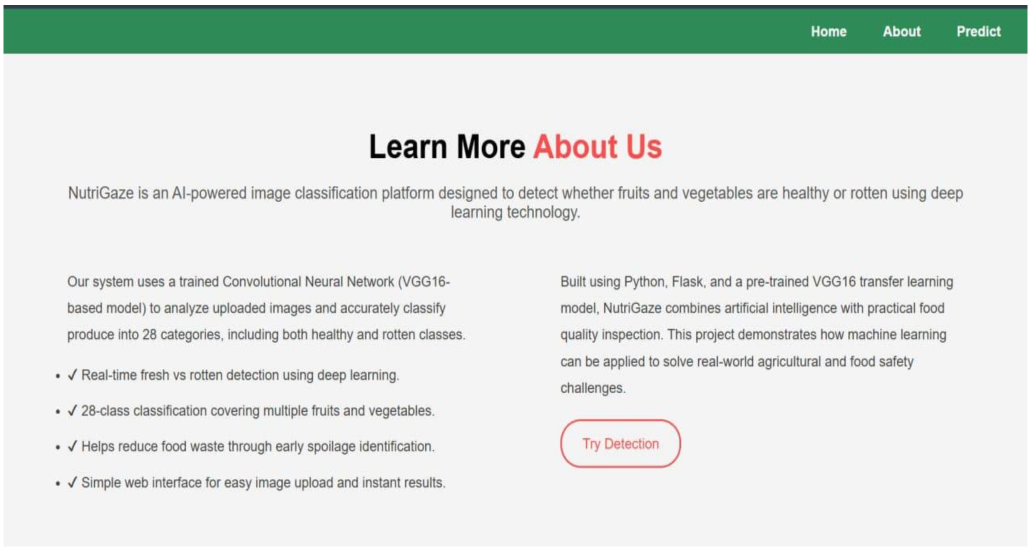
- Currently open access for demonstration purposes.
- Future scope: JWT-based authentication for secure healthcare integration.

9. User Interface

Home Page of Smart sorting:



About page:



Learn More About Us

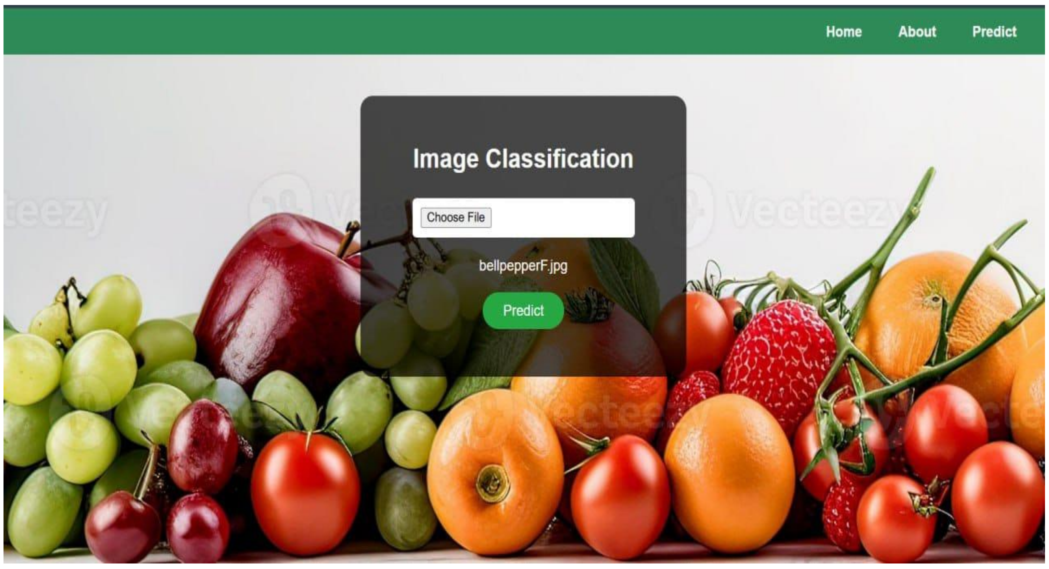
NutriGaze is an AI-powered image classification platform designed to detect whether fruits and vegetables are healthy or rotten using deep learning technology.

Our system uses a trained Convolutional Neural Network (VGG16-based model) to analyze uploaded images and accurately classify produce into 28 categories, including both healthy and rotten classes.

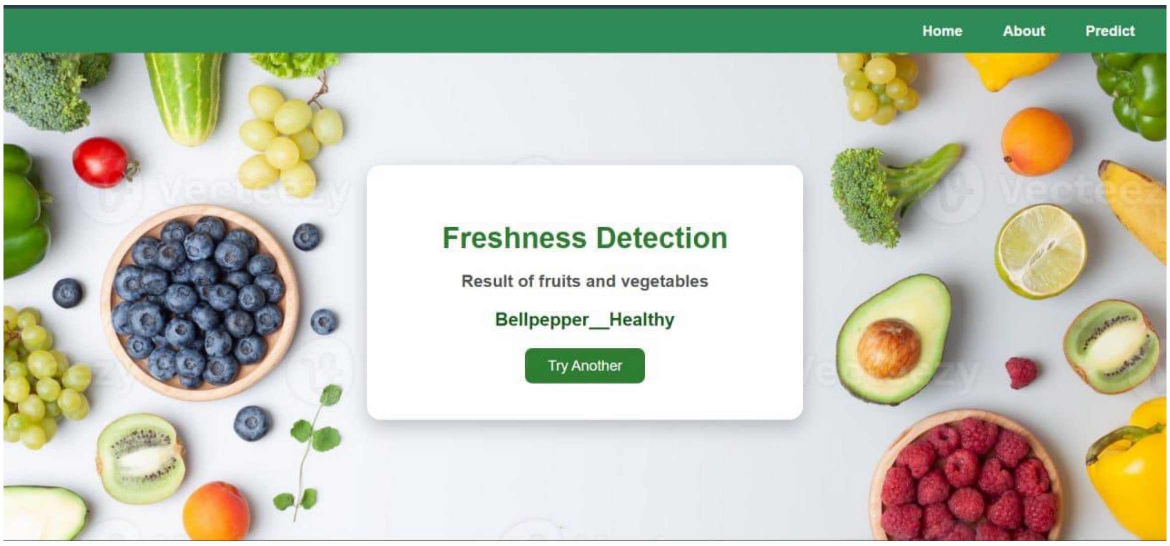
- ✓ Real-time fresh vs rotten detection using deep learning.
- ✓ 28-class classification covering multiple fruits and vegetables.
- ✓ Helps reduce food waste through early spoilage identification.
- ✓ Simple web interface for easy image upload and instant results.

Built using Python, Flask, and a pre-trained VGG16 transfer learning model, NutriGaze combines artificial intelligence with practical food quality inspection. This project demonstrates how machine learning can be applied to solve real-world agricultural and food safety challenges.

Image upload page:



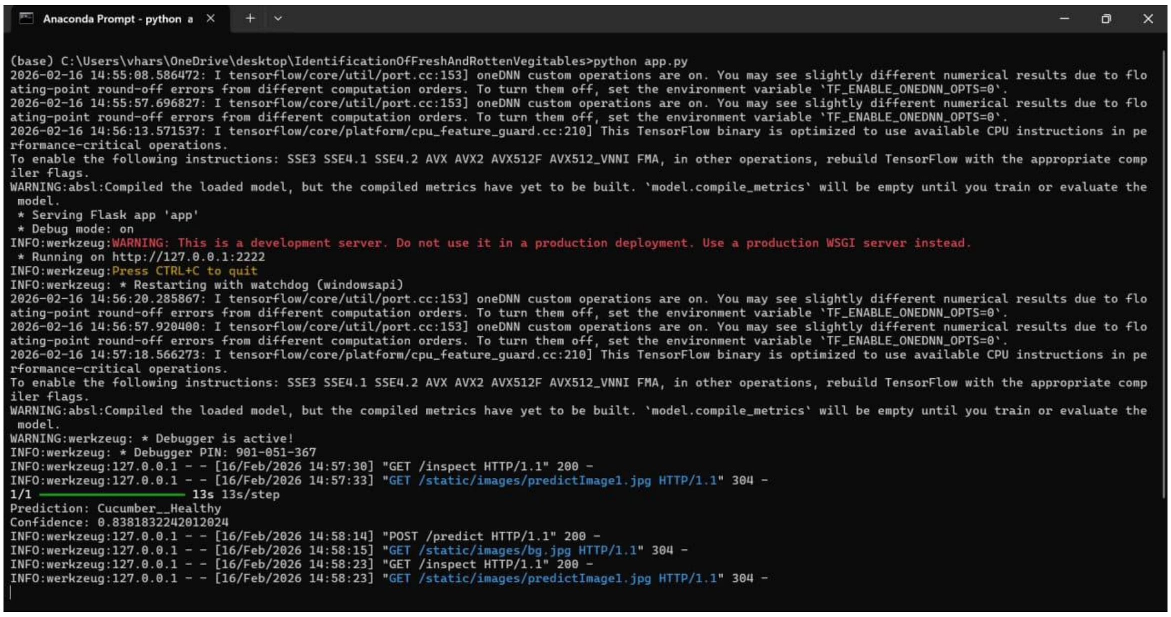
Result page:



10. Testing

• Unit testing for ML model predictions.

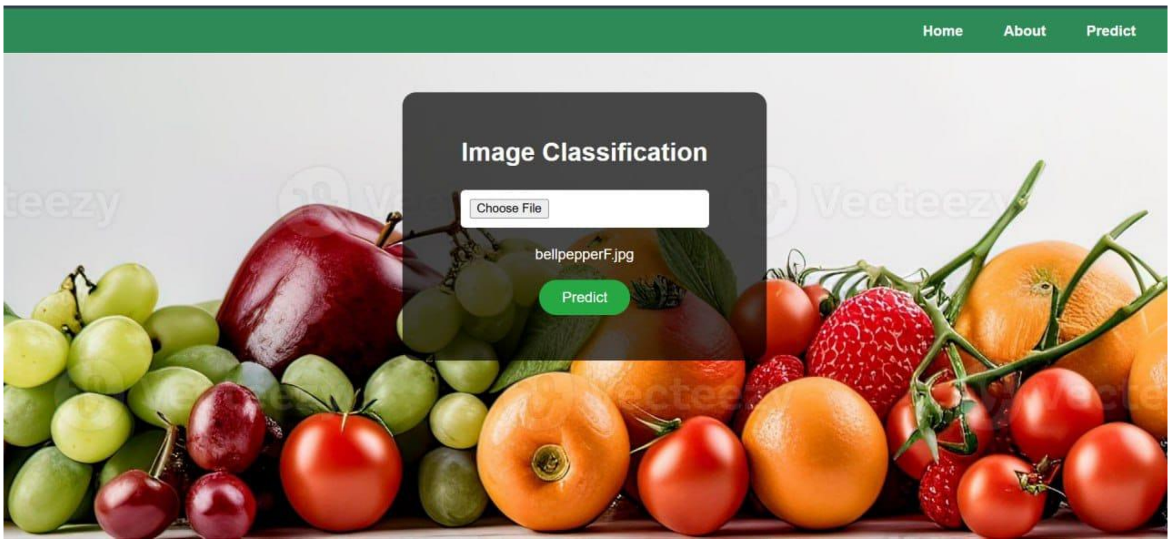
• Functional testing for input validation and dashboard navigation.



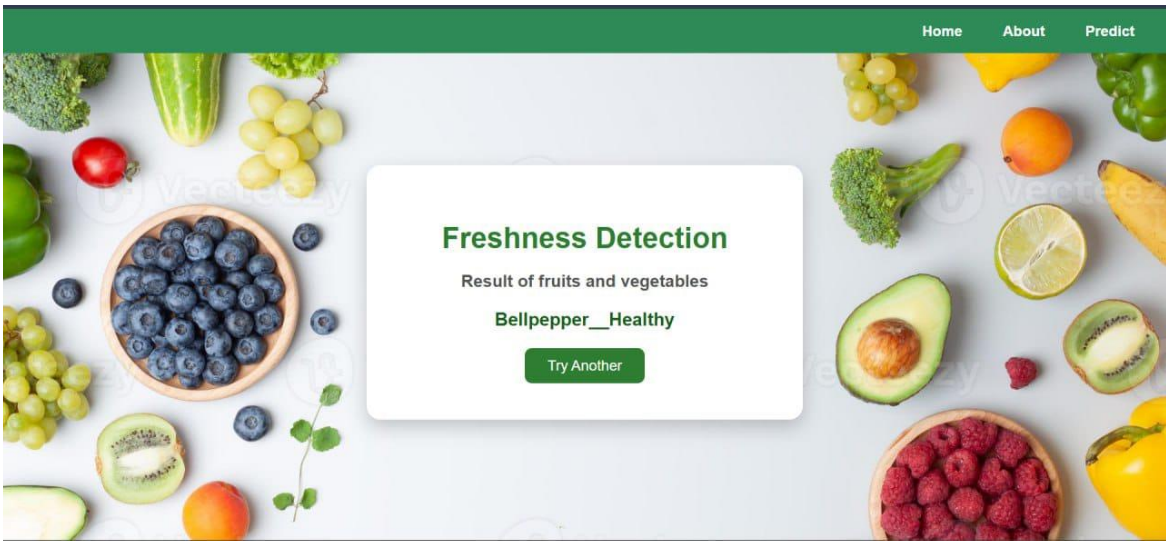
```
(base) C:\Users\vhars\OneDrive\desktop\IdentificationOfFreshAndRottenVegitables>python app.py
2026-02-16 14:55:08.586472: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to flo
ating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2026-02-16 14:55:57.696827: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to flo
ating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2026-02-16 14:56:13.571537: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in pe
rformance-critical operations.
To enable the following instructions: SSE3 SSE4.1 SSE4.2 AVX AVX2 AVX512F AVX512_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate comp
iler flags.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the
 model.
 * Serving Flask app 'app'
 * Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:2222
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with watchdog (windowsapi)
2026-02-16 14:56:20.285867: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to flo
ating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2026-02-16 14:56:57.920400: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to flo
ating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2026-02-16 14:57:18.566273: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in pe
rformance-critical operations.
To enable the following instructions: SSE3 SSE4.1 SSE4.2 AVX AVX2 AVX512F AVX512_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate comp
iler flags.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the
 model.
WARNING:werkzeug: * Debugger is active!
INFO:werkzeug: * Debugger PIN: 901-051-367
INFO:werkzeug:127.0.0.1 - - [16/Feb/2026 14:57:30] "GET /inspect HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [16/Feb/2026 14:57:33] "GET /static/images/predictImage1.jpg HTTP/1.1" 304 -
1/1 ━━━━━━━━━━━━━━━━━━━━ 13s 13s/step
Prediction: Cucumber__Healthy
Confidence: 0.8381832242012024
INFO:werkzeug:127.0.0.1 - - [16/Feb/2026 14:58:14] "POST /predict HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [16/Feb/2026 14:58:15] "GET /static/images/bg.jpg HTTP/1.1" 304 -
INFO:werkzeug:127.0.0.1 - - [16/Feb/2026 14:58:23] "GET /inspect HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [16/Feb/2026 14:58:23] "GET /static/images/predictImage1.jpg HTTP/1.1" 304 -
```

11. Screenshots or Demo

• Vegetable or Fruit image upload interface**.**



• Classification result screen showing prediction.



12. Known Issues

- Limited dataset size may affect generalization.
- Performance may vary with low-quality or blurred images.
- No authentication mechanism currently implemented.

13. Future Enhancements

- Deploy on cloud (AWS/GCP/Azure).
- Add JWT authentication for secure access.
- Expand dataset with more blood cell categories.
- Add real-time microscope camera integration.
- Improve visualization with advanced performance metrics dashboard.

Conclusion

The *Fruits and Vegetables Freshness Detection System* successfully demonstrates the application of **Deep Learning and Transfer Learning** techniques to solve a real-world food quality problem. By leveraging the **VGG16 pre-trained CNN model**, the system is able to classify fruits and vegetables into **28 categories (Healthy and Rotten classes)** with effective accuracy.

The project integrates image preprocessing, data augmentation, model fine-tuning, and a Flask-based web interface to provide an end-to-end solution. Users can upload an image through a simple interface, and the system processes the image, performs prediction, and displays the classification result along with a confidence score.

This solution contributes to:

- Reducing food waste through early spoilage detection
- Assisting vendors, farmers, and consumers in quality assessment
- Demonstrating practical implementation of transfer learning in computer vision