Artificial Intelligence & Machine Learning

Project Documentation

1.Introduction:

- *Project Title*:

Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables

Team Members:

Leader: Bollepalli Teja [228X1A4210]
Team member: G Bharathdwaj [228X1A4226]
Team member: katakam Brahma Rao [228X1A4263]
Team member: Dugyala Swathi kiran [228X1A4254]

2.Project Overview:

- *Purpose:*

The **primary purpose** of the Smart Sorting system using **transfer learning** is to **automate the detection and separation of rotten fruits and vegetables** from fresh ones with high accuracy and efficiency. This is achieved by leveraging pre-trained deep learning models that can quickly learn to identify visual cues of spoilage.

Goals of Smart Sorting System:

1. **Accurate Classification of Produce**
   - Identify and classify fruits and vegetables as *fresh* or *rotten* using deep learning-based image analysis with high precision and recall.
2. **Develop a Transfer Learning-Based Model**
   - Utilize pre-trained models (e.g., ResNet, MobileNet, EfficientNet) and fine-tune them on a domain-specific dataset of fruits and vegetables to save time and computational resources.
3. **Real-Time Detection and Sorting**
   - Achieve real-time or near real-time inference speeds to enable deployment on sorting lines or conveyor belts without delays.
4. **Minimize Manual Inspection**
   - Reduce human error and labor dependency by automating the sorting process with consistent, repeatable performance.
5. **Scalability and Deployment Readiness**

- Design a system that can be scaled and adapted across different environments — from small farms to large warehouses and distribution centers.

6. **Improve Food Supply Chain Efficiency**
    - Prevent rotten produce from contaminating fresh stock, thus improving shelf life and reducing supply chain losses.
7. **Support for Multiple Fruit and Vegetable Types**
    - Extend the system's capability to support classification across various types of fruits and vegetables with minimal retraining.
8. **User-Friendly Interface and Reporting**
    - Provide interfaces for operators to monitor classification results, flag errors, and generate reports for quality assurance.

• *Features:*

1. **Transfer Learning-Based Classification**
   Utilizes powerful pre-trained deep learning models (e.g., MobileNet, ResNet, EfficientNet) fine-tuned for identifying signs of spoilage in fruits and vegetables.

2. **High Accuracy & Robust Detection**
   Capable of detecting various degrees of spoilage (e.g., bruising, mold, discoloration) under different lighting and background conditions.

3. **Multi-Class and Binary Classification Support**
    - **Binary classification**: Fresh vs. Rotten
    - **Multi-class classification**: Fresh, Slightly Spoiled, Severely Rotten

4. **Real-Time Image Processing**
   Processes input from cameras in real-time, enabling on-the-fly sorting with minimal latency.

5. **Edge and Cloud Compatibility**
   Can be deployed on edge devices (e.g., Raspberry Pi, NVIDIA Jetson) or integrated with cloud platforms for centralized processing.

Core Functionalities:-

- **Image Capture & Preprocessing**
  Captures images of fruits/vegetables and preprocesses them (resizing, normalization) for model input.

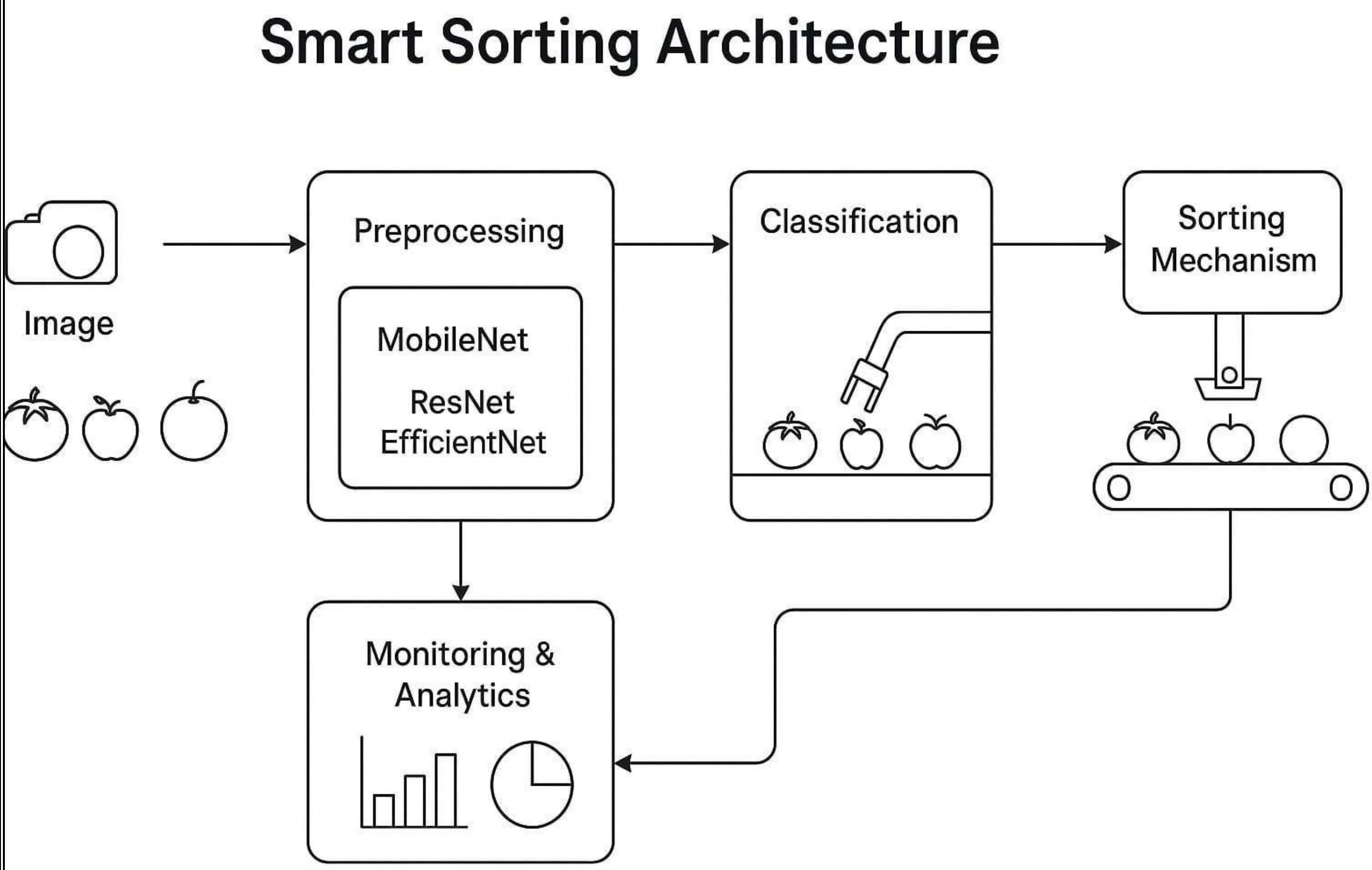
- **Spoilage Detection Using Deep Learning**
  Predicts freshness/spoilage based on visual patterns using trained CNN models.

- **Sorting Decision Engine**
  Decides the routing action (e.g., keep, discard, alert) based on prediction probabilities.

- **Model Training & Fine-Tuning Module**
  Provides tools to retrain and improve model accuracy using new or additional datasets.

3. Architecture:

The architecture is designed to automatically detect and sort rotten fruits and vegetables using machine vision and deep learning, particularly through transfer learning, to improve accuracy and reduce development time.



4. Setup Instructions:

Prerequisites:

Hardware:

- Camera (USB, PiCam, or webcam)
- Raspberry Pi / Jetson Nano / Laptop with GPU
- Actuator or Servo Motor (for sorting)
- Breadboard, jumper wires (if using microcontrollers)
- Power supply

Software:

- Python 3.8+
- TensorFlow / Keras
- OpenCV
- Numpy, Matplotlib
- scikit-learn
- RPi.GPIO (for Raspberry Pi sorting)
- Jupyter Notebook or any IDE (VSCode, Thonny)

5. Folder Structure:

```
smart_sorting_project/
|
├── 🗁 dataset/
|   ├── 🗁 fresh/
|   |   ├── apple1.jpg
|   |   ├── banana2.jpg
|   |   └── ...
|   └── 🗁 rotten/
|       ├── apple1.jpg
|       ├── banana2.jpg
|       └── ...
|
├── 🗁 models/
|   └── fruit_sorter_model.h5       # Trained model
|
├── 🗁 utils/
|   ├── image_utils.py            # Preprocessing or augmentation scripts
|   └── hardware_control.py        # Servo/GPIO control functions
|
├── 🗁 scripts/
```

```
│   ├── data_preprocessing.py       # Data loading and augmentation
│   ├── train_model.py              # Model training script
│   ├── live_inference.py           # Real-time prediction with camera
│   └── sort_with_servo.py          # Sorting control based on predictions
│
├── 🗀 notebooks/
│   └── exploration.ipynb           # Jupyter notebook for initial testing
│
├── 🗀 config/
│   └── config.yaml                 # Settings like camera source, thresholds, paths
│
├── 🗀 logs/
│   └── training_log.csv            # Model training logs and accuracy reports
│
├── requirements.txt                # Python dependencies
├── README.md                       # Project overview and instructions
└── setup_instructions.pdf          # (Optional) PDF guide for setup
```

6. Running the Application:

- Dataset is organized in dataset/fresh/ and dataset/rotten/
- Model is trained and saved as models/fruit_sorter_model.h5
- Camera and (if needed) servo hardware are properly connected
- All dependencies installed:
    - bash
    - pip install -r requirements.txt

7. API Documentation:

This RESTful API enables:

- Uploading fruit/vegetable images
- Classifying the image (Fresh or Rotten)
- Retrieving model prediction results
- Optional: triggering a hardware sorting action

8. Testing:

Smart Sorting System – API Testing Guide

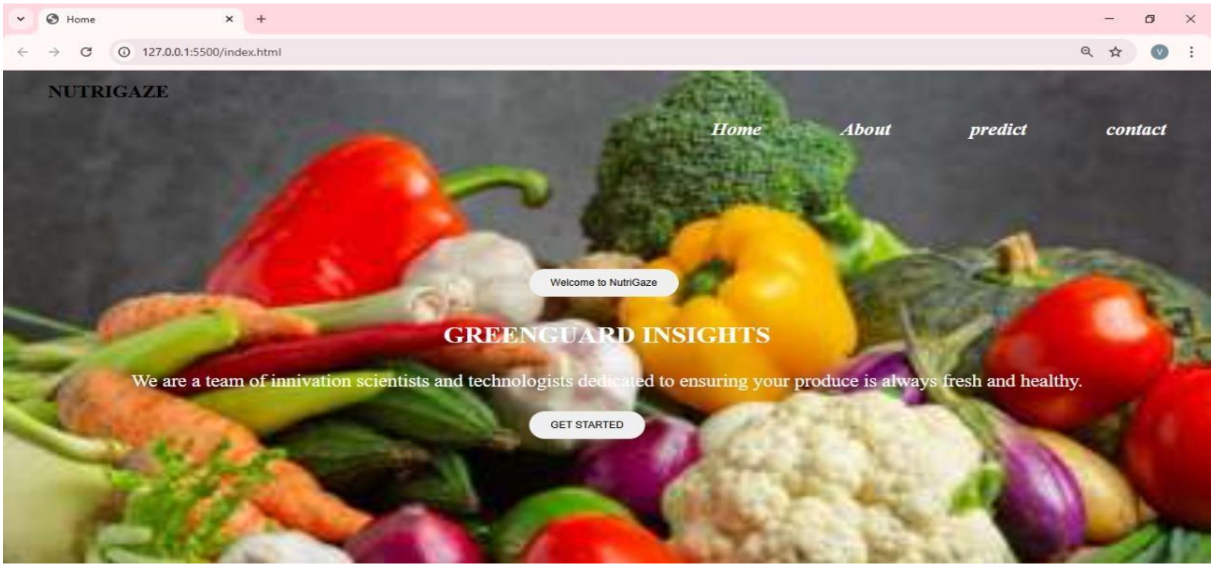
Tools You Can Use:

- Postman (GUI-based testing)
- cURL (Command-line)
- Python requests module (automated test scripts)
- Unit testing in Flask with pytest or unittest

9. Screenshots:

Templates screenshots:

1. Home Page:



2. About Page:

3. Predict Page:

10. Known Issues:-

Many studies focus on a small number of fruits and vegetables often excluding vegetables entirely and use clean, controlled settings with white or simple backgrounds .

Training data frequently lacks variability in lighting, occlusion, multi-object scenes, and cluttered background even though this is typical in practical applications .

When some categories have very few samples, models may bias toward majority classes unless proper balancing and augmentation are used.

11. Future Enchancements:-

Incorporate a wider variety of fruits and vegetables captured under diverse lighting, backgrounds, and occlusions to improve model generalization in real-world environments.

Develop and deploy efficient deep learning models (e.g., MobileNet, TinyYOLO) optimized for edge devices to enable real-time sorting in farms, markets, and low-power settings.

Extend the system to handle multiple items per frame and use segmentation techniques to detect specific rotten areas, enhancing precision in mixed or cluttered scenes.