PROJECTTITLE

SMART SORTING: Transfer Learning for Identifying Rotten Fruits and Vegetables

Artificial Intelligence & Machine Learning

TEAM ID: LTVIP2025TMID36245

TEAM MEMBERS(ROLES):

1. Chittiboyina Muni nandini– (Team Leader)
2. Bodimani Yagna Deepika (Team Member)
3. Yebbili Nagaraju (Team Member)
4. Puram Navya (Team Member)

Table of Contents

1. Introduction

Welcome to the **"Smart Sorting Prediction App,"** also known as **"SMART SORTING: Transfer Learning For Identifying Rotten Fruits And Vegetables"**! This project is designed to tackle a common problem in households and industries: efficiently identifying whether a fruit or vegetable is fresh or rotten.

At its core, this application leverages the power of Deep Learning, specifically Transfer Learning, to achieve highly accurate predictions. We utilize the pre-trained VGG16 Convolutional Neural Network, a robust and widely used model, and fine-tune it on a specialized dataset containing images of both healthy and rotten produce. This approach allows us to benefit from the extensive knowledge VGG16 gained from millions of images, adapting it quickly to our specific task.

The system is deployed as a user-friendly Flask web application. This means you can interact with it through a simple web interface in your browser. Users can easily upload an image of a fruit or vegetable, and the backend, powered by our trained deep learning model, will process the image and provide an instant prediction—for example, **"Apple_healthy"** or **"Strawberry_rotten,"** along with a confidence score.

This project aims not only to provide a practical sorting solution but also to serve as a comprehensive learning experience, covering aspects from data handling and model training to full-stack web deployment.

2. Project Overview

The "Smart Sorting Prediction App" is an innovative full-stack web application designed to revolutionize quality control in the agricultural and food industries. Its core mission is to intelligently identify and classify fruits and vegetables as either healthy or rotten using advanced image recognition capabilities powered by machine learning. This project addresses a critical challenge in the supply chain:The efficient and accurate sorting of produce to minimize waste, enhance product quality, and ensure that only fresh goods reach consumers.

The agricultural sector frequently grapples with inefficiencies in manual sorting, which can be time-consuming, prone to human error, and inconsistent. This often leads to significant post-harvest losses due to undetected spoilage, impacting profitability and contributing to food waste. The Smart Sorting Prediction App aims to mitigate these issues by providing an automated, precise, and rapid solution for quality assessment.

PURPOSE:

The primary purpose of this project is to develop and demonstrate a practical application of deep learning and web development for a real-world problem. It serves as a proof-of-concept for how machine vision can be integrated into everyday processes to deliver tangible benefits. Specifically, the application aims to:

Automate Quality Assessment: Provide a tool for rapid, consistent, and objective evaluation of fruit and vegetable health.

Reduce Food Waste: By accurately identifying rotten produce early, it helps in preventing the spoilage of entire batches and optimizes inventory management.

Improve Efficiency: Streamline the sorting process, reducing the need for extensive manual inspection.

FEATURES:

The "Smart Sorting Prediction App" is built with user experience and robust functionality in mind, offering several key features:

- **User-Friendly Web Interface**: An intuitive and clean web-based interface allows users to easily upload images of various fruits and vegetables directly through their browser.

Real-time Prediction: The application provides near real-time analysis of uploaded images, classifying the produce as "healthy" or "rotten" almost instantly.

- **Deep Learning Powered Core:** At its heart, the application utilizes a deep learning model, specifically leveraging the power of Transfer Learning. This approach allows the model to achieve high accuracy in classification by adapting knowledge from large, pre-trained neural networks to the specific task of identifying rotten produce. The trained model, healthy_vs_rotten.h5, is seamlessly integrated into the backend.
- **Prediction Confidence Levels:** Alongside the classification, the application displays a confidence score, giving users an indication of the model's certainty in its prediction.
- **Scalable Architecture:** Built with a Python Flask backend and a lightweight HTML/CSS/JavaScript frontend, the application has a modular design that facilitates future enhancements and potential scaling.

This project showcases a comprehensive solution, from data processing and model training to web application deployment, illustrating the power of integrating machine learning into practical full-stack development. It stands as a testament to leveraging AI for more sustainable and efficient practices in the agricultural domain.

3. Architecture

- **Frontend**: The frontend is built using standard web technologies: HTML for structure, CSS for styling, and JavaScript for interactive elements. It consists of static templates rendered by the backend to provide the user interface for image uploads and displaying prediction results.
- **Backend**: The backend is developed with Python using the Flask web framework. It handles incoming image uploads, preprocesses the images, performs inference using the loaded machine learning model (healthy_vs_rotten.h5), and sends the prediction results back to the frontend.
- **Database**: This project does not utilize a persistent database (like MongoDB). All image processing and predictions are handled in-memory by the Flask application, and results are displayed directly to the user.

4. Setup Instructions

Prerequisites:

- Python 3.x
- Pip (Python package installer)
- Required Python libraries (e.g., Flask, TensorFlow/Keras, OpenCV, Pillow, NumPy)
- Google Colab (recommended for running due to GPU requirements for ML models)
- Ngrok (for public access to the Colab-hosted application)

Installation:

Clone the repository:

Navigate into the project directory:

cd SmartSortingPredictionApp-

Install Python dependencies:

pip install Flask tensorflow opencv-python Pillow numpy (or list specific versions if needed in a requirements.txt file)

Download the dataset and pre-trained model:

The healthy_vs_rotten.h5 model is included via Git LFS. Ensure Git LFS is installed and configured (git lfs pull).

The dataset (if needed for re-training or local setup) should be downloaded separately (e.g., from Kaggle, as mentioned in my demo video).

5. Folder Structure

Client (Frontend):

templates/: Contains HTML files that define the user interface (e.g., index.html).

static/: (Expected if present) Would contain static assets like css/ for stylesheets, js/ for client-side JavaScript, and uploads/ for temporary uploaded images.

Server (Backend):

app.py: The main Flask application file, handling routes, model loading, image processing, and predictions.

train_model.py: Script for training and evaluating the machine learning model.

data_prep_and_viz.py: Script for data preprocessing and visualization.

healthy_vs_rotten.h5: The pre-trained Keras/TensorFlow model file used for predictions.

6. Running the Application

Local Setup (e.g., in a terminal after cloning):

1. Ensure all prerequisites are met and dependencies are installed.
2. Start the Flask backend: python app.py
3. The application will typically run on http://127.0.0.1:5000.

Google Colab Setup (as shown in the demo video i.e., GITHUB repositary):

1. Mount Google Drive.
2. Unzip your project into /content/SmartSortingProject.
3. Change the current working directory to /content/SmartSortingProject.
4. Install required libraries within the Colab notebook.
5. Run app.py in the background (e.g., !nohup python app.py > app.log 2>&1 &).
6. Start ngrok to expose the Flask port (e.g., !ngrok http 5000). Access the application via the provided ngrok public URL.

7. API Documentation

The application provides a web interface rather than a public API for direct consumption. However, the backend exposes the following key routes handled by

app.py:

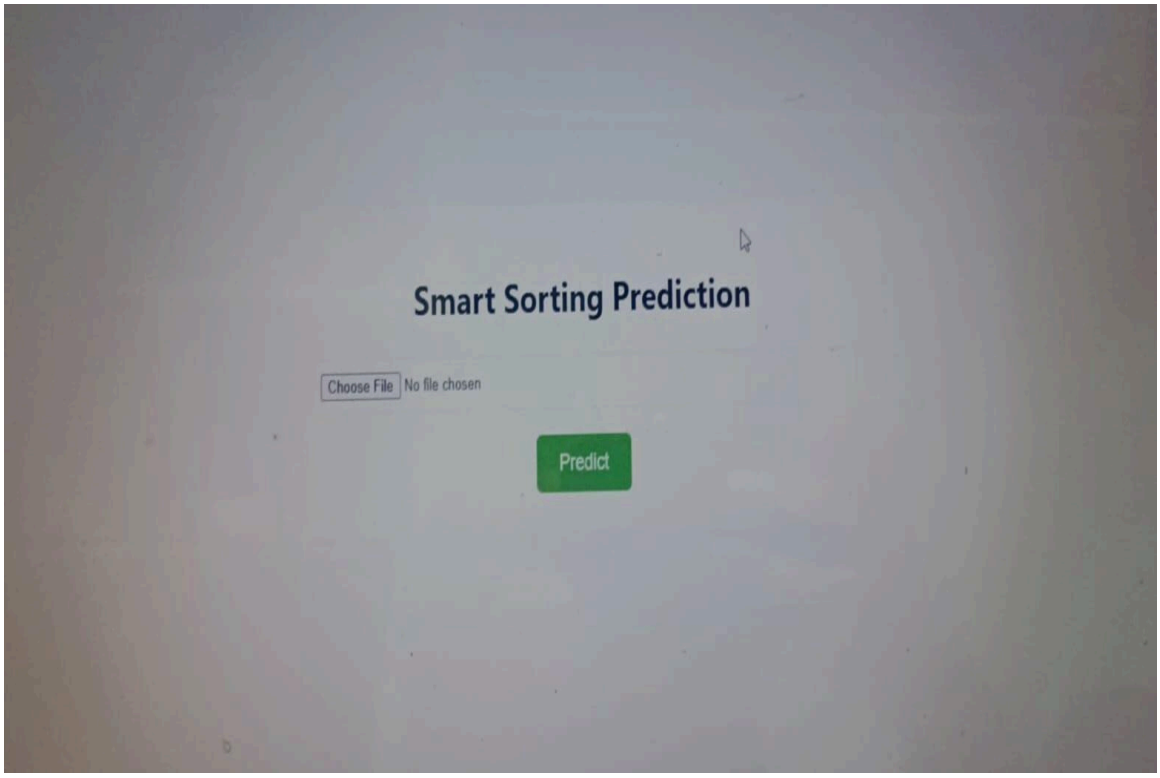
/ (GET): Serves the main index.html page, which is the entry point for the application.

/predict (POST): Accepts an image file (typically via a form submission). It processes the image, runs it through the healthy_vs_rotten.h5 model, and returns the prediction result (e.g., "Healthy" or "Rotten" along with confidence scores).

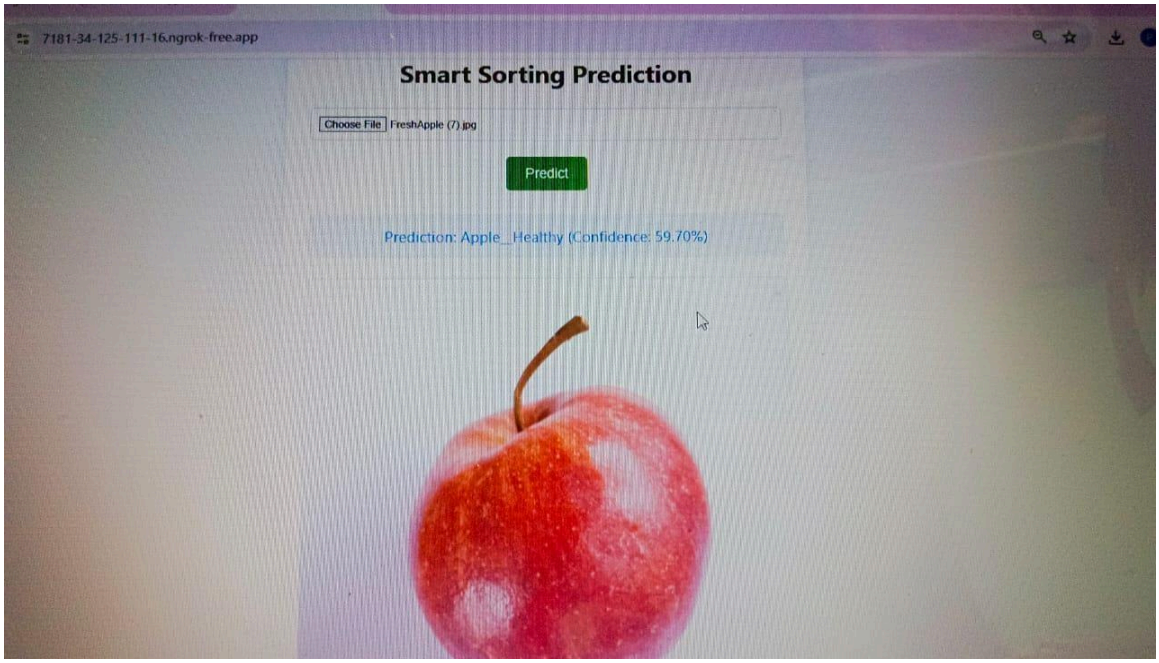
8. User Interface (Screenshots Section)

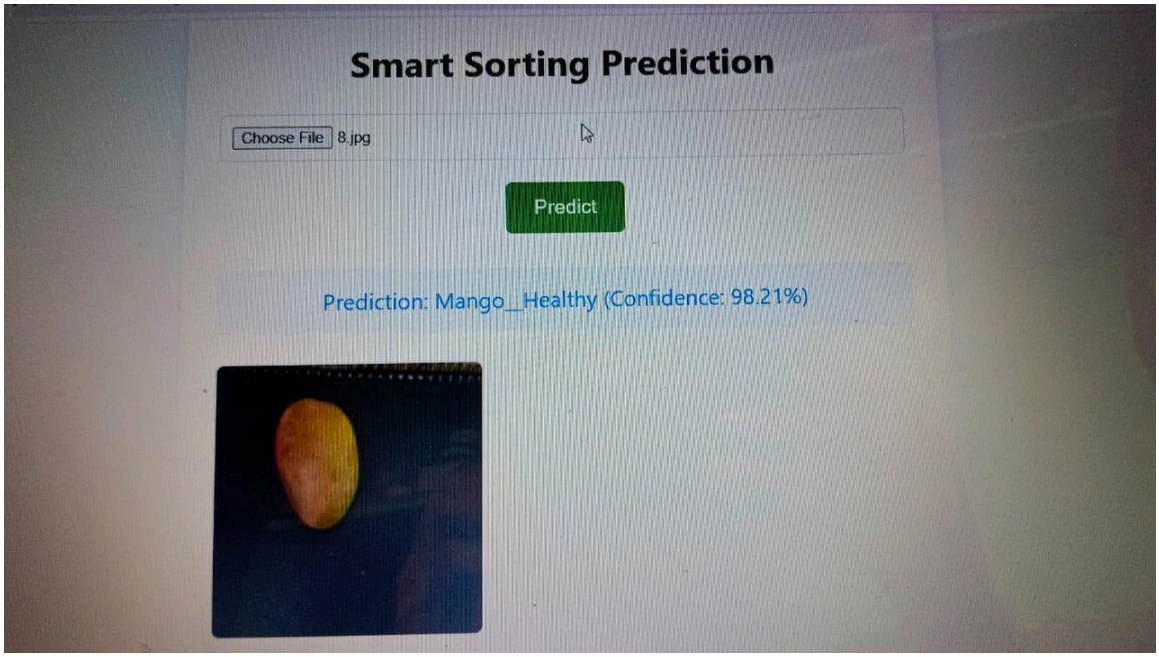
The UI is a simple web page allowing users to upload an image file. After processing, it displays the prediction result clearly.

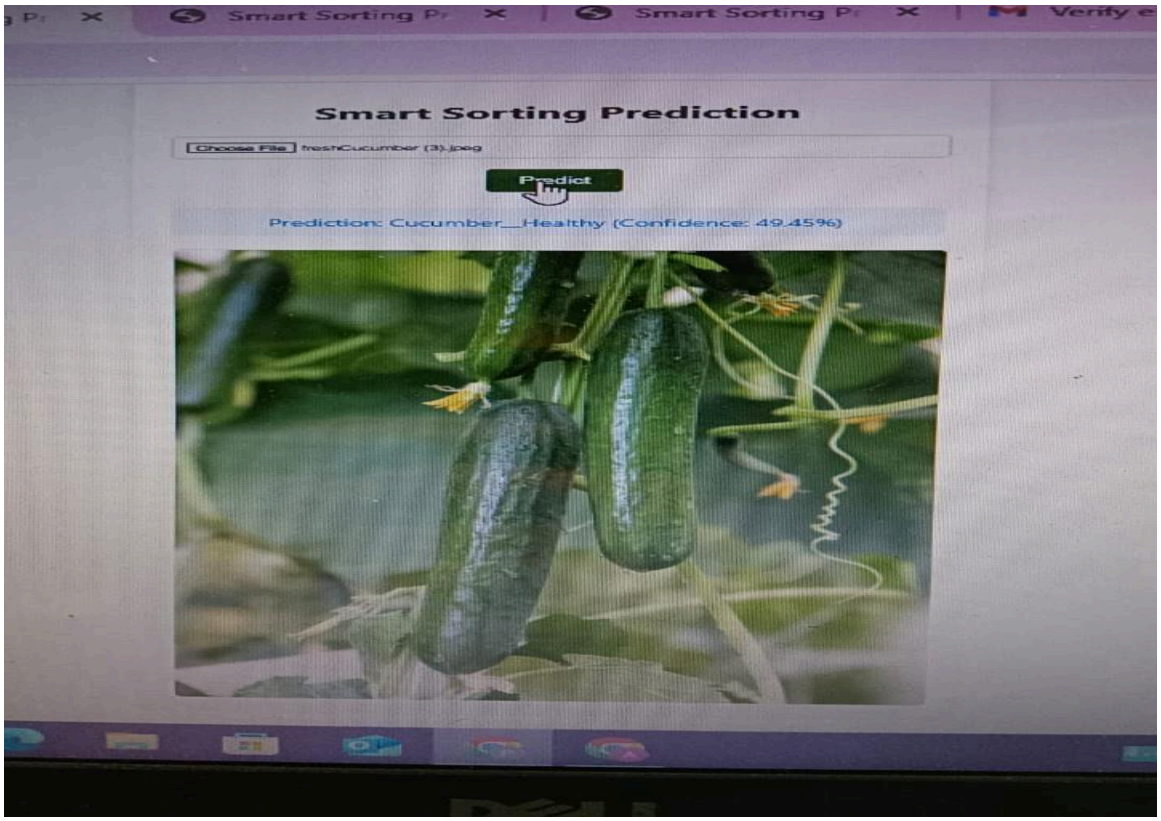
THE WEB PAGE ALLOWING USERS TO UPLOAD AN IMAGE FILE:

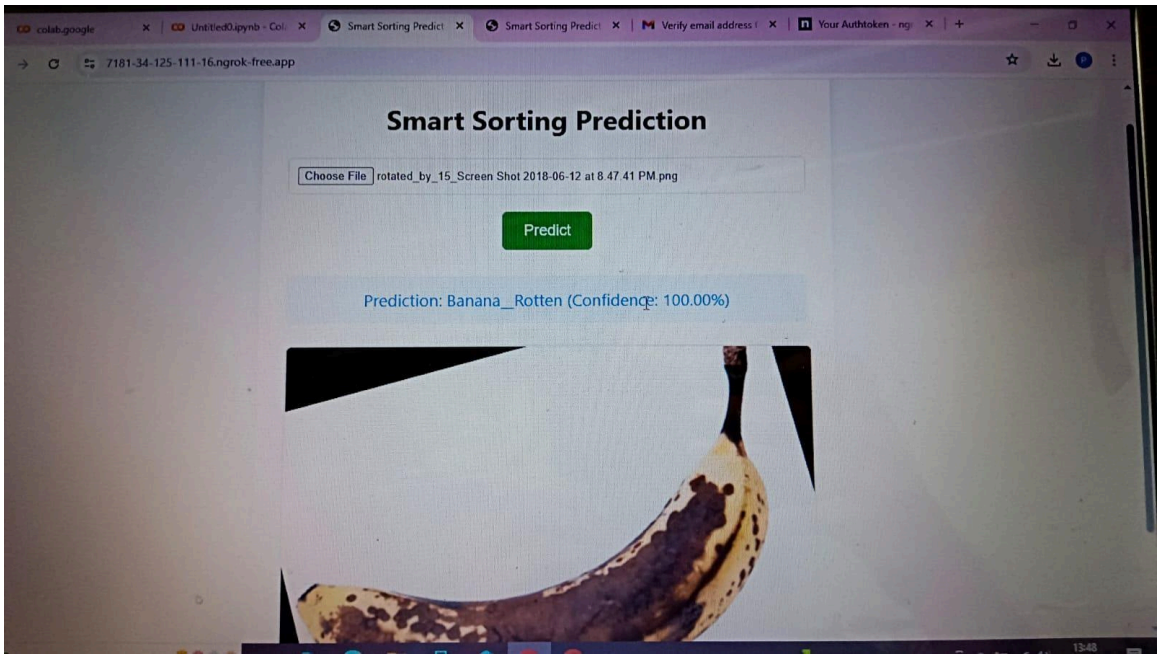


PREDICTION RESULTS:


7181-34-125-111-16.ngrok-free.app
Smart Sorting Prediction
Choose File
FreshApple (7).jpg
Predict
Prediction: Apple__Healthy (Confidence: 59.70%)


Smart Sorting Prediction
Choose File
8.jpg
Predict
Prediction: Mango__Healthy (Confidence: 98.21%)


Smart Sorting Prediction
Choose File freshCucumber (3).jpeg
Predict
Prediction: Cucumber__Healthy (Confidence: 49.4596)


7181-34-125-111-16.ngrok-free.app
Smart Sorting Prediction
Choose File rotated_by_15_Screen Shot 2018-06-12 at 8.47.41 PM.png
Predict
Prediction: Banana__Rotten (Confidence: 100.00%)

9. TESTING:

LINK **Strategy**: Testing primarily involves verifying the accuracy of the machine learning model (during training in train_model.py) and ensuring the web application correctly handles image uploads and displays predictions.

Tools: Standard Python unit testing (e.g., unittest or pytest) could be used for backend logic, and manual testing for the web interface.

10. DEMO VIDEO :

11. KNOWN ISSUES:

Model Generalization: The model's performance might vary with images of fruits/vegetables not well-represented in the training dataset or under different lighting/background conditions.

Scalability: The current setup is for demonstration; scaling to handle many concurrent users would require more robust deployment (e.g., using Gunicorn/Nginx) and potentially cloud resources.

12. FUTURE ENHANCEMENTS:

- **Expand Dataset and Model:** Train on a larger, more diverse dataset to recognize more types of fruits/vegetables and a wider range of diseases/spoilage.
- **User Accounts:** Implement user authentication to manage user-specific data or features.
- **Prediction History:** Add a database to store user upload history and prediction results.
- **Mobile Responsiveness:** Improve the frontend design for better display on mobile devices.

- **Real-time Feedback:** Provide more detailed feedback to the user on image processing or model inference time.
- **Deployment Automation:** Automate deployment to a cloud platform (e.g., Google Cloud Run, AWS Elastic Beanstalk