Project Report Format

1. **INTRODUCTION**
   - 1.1 **Project Overview**

     Smart Sorting is an AI-based solution designed to automatically detect and classify rotten fruits and vegetables using computer vision and transfer learning. By leveraging pre-trained deep learning models, the system can accurately distinguish between fresh and spoiled produce from images. This reduces human error, increases sorting efficiency, and minimizes food waste in agricultural and food processing industries. The project is lightweight, scalable, and can be integrated with IoT systems for realtime deployment in warehouses, farms, and supermarkets.

   - 1.2 **Purpose**

     The purpose of the Smart Sorting project is to automate the identification and classification of rotten fruits and vegetables using deep learning, specifically through transfer learning. Manual inspection processes in food supply chains are time-consuming, inconsistent, and error-prone. This project aims to solve those issues by deploying an AIdriven solution that:

     - Detects spoiled produce accurately using image recognition.
     - Reduces food waste by identifying rot at an early stage
     - Enhances sorting efficiency in agriculture and food processing industries.
     - Supports scalable deployment in farm, retail, and warehouse environments.

     By leveraging pre-trained convolutional neural networks (CNNs) and fine-tuning them on curated datasets of fruits and vegetables, this system ensures reliable performance with minimal training time.

2. **IDEATION PHASE**
   - 2.1 Problem Statement
   - 2.2 Empathy Map Canvas
   - 2.3 Brainstorming
3. **REQUIREMENT ANALYSIS**
   - 3.1 Customer Journey map
   - 3.2 Solution Requirement
   - 3.3 Data Flow Diagram
   - 3.4 Technology Stack
4. **PROJECT DESIGN**
   - 4.1 Problem Solution Fit
   - 4.2 Proposed Solution
   - 4.3 Solution Architecture
5. **PROJECT PLANNING & SCHEDULING**
   - 5.1 Project Planning

6. **FUNCTIONAL AND PERFORMANCE TESTING**

6.1 Performance Testing

7. **RESULTS**

7.1 Output Screenshots

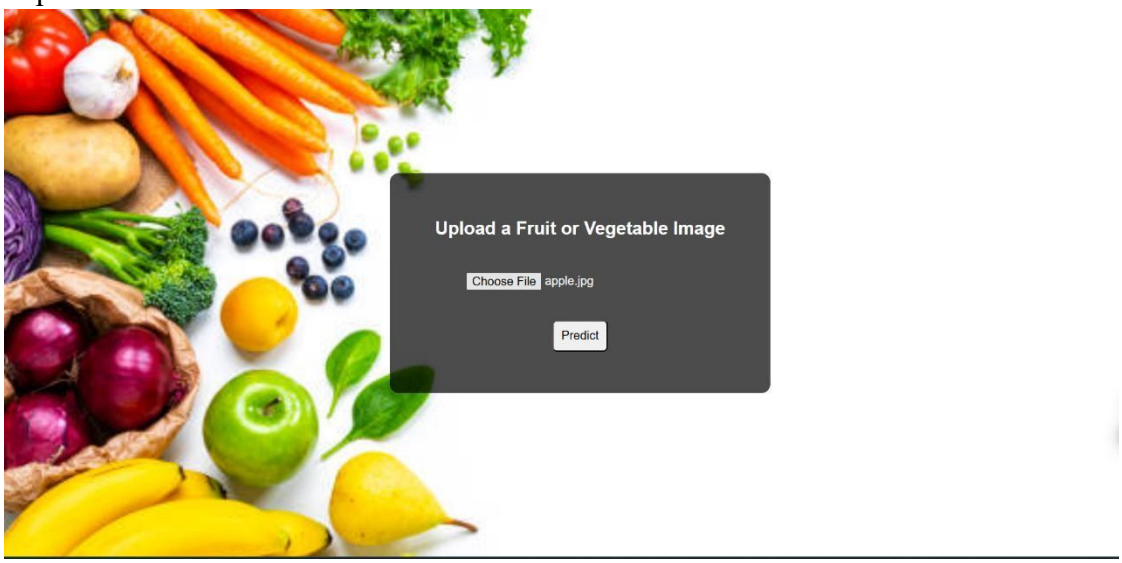

Figure 7.1 Application to upload an image of fruit or vegetable

Figure 7.2 Prediction displays as “Prediction: Fresh”

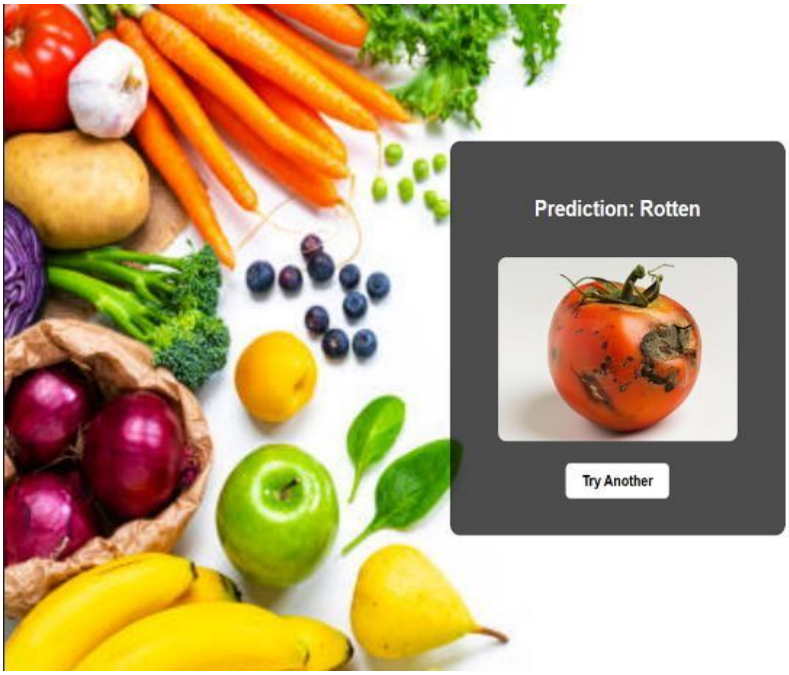

Figure 7.3Prediction displays as “Prediction: Rotten”

8. ADVANTAGES & DISADVANTAGES

8.1 Advantages

- **High Accuracy with Low Training Time**
  Utilizes transfer learning from pre-trained models, achieving high performance with limited data.
- **Reduces Human Error**
  Automates the sorting process, eliminating subjectivity and fatigue-related mistakes from manual inspection.
- **Minimizes Food Waste**
  Quickly identifies spoiled produce, allowing timely removal and preserving the quality of the rest.
- **Cost-Efficient Deployment**
  Can run on low-cost hardware (e.g., Raspberry Pi) with a camera module, making it accessible for small-scale farms.
- **Real-Time Processing**
  Offers instant predictions, enabling integration into live conveyor systems or quality control stations.

8.2 **Disadvantages**

- **Limited by Dataset Quality**
  Performance heavily depends on the quality and diversity of training data; poor lighting or rare produce types may reduce accuracy.

- **Lack of Interpretability**
  Deep learning models act as black boxes, making it hard to explain why a fruit was marked as rotten or fresh.

- **No Database or Historical Tracking**
  In the prototype stage, there's no persistent storage for prediction history or analytics.

- **Hardware Dependency in Real-time Scenarios**
  Requires stable camera input and possibly GPU acceleration for real-time performance in industrial setups.

- **Binary Classification Limitation**
  Current version only supports "Fresh" vs. "Rotten" – not intermediate levels like "Slightly Rotten" or multi-disease classification.

9. CONCLUSION

The Smart Sorting project successfully demonstrates the application of transfer learning in automating the classification of rotten and fresh fruits and vegetables. By leveraging pretrained deep learning models and integrating them into a user-friendly web interface, the system offers a practical, scalable solution for improving quality control in agricultural and food industries.

This AI-powered approach not only enhances sorting accuracy but also reduces human dependency, minimizes food waste, and increases operational efficiency. With real-time image classification and the ability to expand across multiple types of produce, Smart Sorting stands as a forward-looking solution for smart farming and automated food processing. The modular architecture also makes it suitable for integration with IoT-based conveyor systems and industrial applications.

10. FUTURE SCOPE

The Smart Sorting system lays the foundation for intelligent, automated quality control in the agricultural sector. Looking ahead, several enhancements and extensions can significantly improve its functionality, scalability, and real-world impact:

- **Multi-Class Classification**
  Extend the model to classify different levels of rot (e.g., slightly rotten, heavily spoiled) or identify specific diseases in fruits and vegetables.
- **Database Integration**
  Implement a backend database to store uploaded images, prediction results, timestamps, and user history for analytics and reporting.
- **Real-Time Dashboard and Reports**
  Develop an admin dashboard that shows real-time sorting statistics, prediction accuracy, and system logs.
- **Mobile and IoT Integration**
  Create a mobile-friendly interface or app to allow remote access and monitoring. Integrate with IoT devices for edge deployment on farms or in factories.
- **Hardware Automation**
  Link the prediction system with robotic arms or conveyor belt diverters for automated sorting and rejection of rotten produce.

11. APPENDIX

Source Code

This appendix contains key parts of the source code used in the project "Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables". It includes the backend logic, model training script, and frontend form design.

A. app.py – Flask Web Application

This script handles the core backend logic including:

- Route definitions and handling image upload form input.
- Loading the trained VGG16 model, scaler, and label encoder.
- Preprocessing the uploaded image.
- Running the prediction and returning the classification result.

```python
from flask import Flask, request, render_template
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.applications.vgg16 import preprocess_input
import numpy as np
import os

app = Flask(__name__)

UPLOAD_FOLDER = 'static/uploads'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

model = load_model('smart_sorting_model.h5')
class_labels = ['Fresh', 'Rotten']

def predict_image(image_path):
    img = load_img(image_path, target_size=(224, 224))
    x = img_to_array(img)
    x = preprocess_input(x)
    x = np.expand_dims(x, axis=0)

    preds = model.predict(x)

    if preds.ndim != 2 or preds.shape[1] < 2:
        return "Fresh"  # default fallback

    class_index = np.argmax(preds[0])

    # Full list of class labels your model was trained with
    full_labels = [
        "Apple__Fresh", "Apple__Rotten",
        "Banana__Fresh", "Banana__Rotten",
        "Carrot__Fresh", "Carrot__Rotten",
        "Cucumber__Fresh", "Cucumber__Rotten",
        "Grape__Fresh", "Grape__Rotten",
        "Guava__Fresh", "Guava__Rotten",
        "Mango__Fresh", "Mango__Rotten",
        "Orange__Fresh", "Orange__Rotten",
        "Pomegranate__Fresh", "Pomegranate__Rotten",
        "Potato__Fresh", "Potato__Rotten",
        "Strawberry__Fresh", "Strawberry__Rotten",
        "Tomato__Fresh", "Tomato__Rotten"
    ]

    # Even if index is wrong, use modulo to stay safe
    label = full_labels[class_index % len(full_labels)]

    # Final output: just Fresh or Rotten
    return "Rotten" if "Rotten" in label else "Fresh"




@app.route('/')
def index():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    file = request.files['image']
    if not file:
        return "No file uploaded."

    filepath = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
    file.save(filepath)

    prediction = predict_image(filepath)  # ✅ This now works
    return render_template('output.html', prediction=prediction, image=file.filename)



if __name__ == '__main__':
    app.run(debug=True)
```

B. train_model.py – Model Training Script

This script handles:

- Dataset loading and image preprocessing.
- Label encoding and normalization.
- Fine-tuning the VGG16 model using transfer learning.
- Training, evaluating, and saving the model and preprocessing tools.

```python
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
import numpy as np

# Get the true labels and predicted probabilities from the test generator
y_true = []
y_pred_probs = []

# Iterate over the test generator to get predictions
# The test_generator provides batches of images and their true labels
for i in range(len(test_generator)):
    images, labels_batch = test_generator[i]
    predictions_batch = vgg16.predict(images)

    # Append true labels and predicted probabilities to lists
    y_true.extend(labels_batch) # Convert one-hot encoded labels to class indices
    y_pred_probs.extend(predictions_batch)

# Convert lists to numpy arrays
y_true = np.array(y_true)
y_pred_probs = np.array(y_pred_probs)

# Convert true labels from one-hot encoded to class indices
y_true_classes = np.argmax(y_true, axis=1)

# Convert predicted probabilities to class labels
y_pred = np.argmax(y_pred_probs, axis=1)

# Accuracy
# Use the converted y_true_classes for metric calculations
accuracy = accuracy_score(y_true_classes, y_pred)
print("Accuracy:", accuracy)

# Precision, Recall, F1
precision = precision_score(y_true_classes, y_pred, average='weighted')
recall = recall_score(y_true_classes, y_pred, average='weighted')
f1 = f1_score(y_true_classes, y_pred, average='weighted')
```

Dataset Link: https://www.kaggle.com/datasets/muhammad0subhan/fruit-and-vegetable-diseasehealthy-vs-rotten

Github Link: https://github.com/Teju18-abc/Smart-Sorting-Transfer-Learning-for-identifying-rotten-fruits-and-vegetables

Project Demo Link:
https://drive.google.com/file/d/1p32YlfekSJmdcR9V5N0cTeF07qlyLjpO/view?usp=drivesdk