SMART SORTING ROTTEN FRUITS AND VEGETABLES

1. Introduction

The "Smart Sorting of Rotten Fruits and Vegetables" project is an intelligent system designed to automatically detect and sort rotten fruits and vegetables using image classification with deep learning. This project leverages transfer learning with MobileNetV2 to build an efficient and accurate model that can help reduce food waste, improve quality control, and increase the efficiency of the food processing pipeline.

2. Project File Structure

```
smart_sorting_project/
│
├── code/
│   ├── train_model.py          # Model training script
│   └── predict.py              # Prediction/testing script
│
├── dataset/
│   ├── train/
│   │   ├── fresh/              # Training images of fresh produce
│   │   └── rotten/             # Training images of rotten produce
│   └── test/
│       ├── fresh/              # Testing images of fresh produce
│       └── rotten/             # Testing images of rotten produce
│
├── models/
│   └── fruit_classifier.h5     # Saved trained model
│
├── results/
│   ├── accuracy_plot.png       # Model accuracy and loss graphs
│   └── prediction_samples.png # Sample prediction results
│
├── report/
│   └── Smart_Sorting_Report.pdf  # Project Report (this file)
│
└── README.md
```

---

3. Step-by-Step Process

Step 1: Data Collection

Collect images of fresh and rotten fruits and vegetables.

Organize the dataset into train and test folders with subfolders for fresh and rotten images.

Step 2: Data Preprocessing

Resize all images to 224x224 pixels.

Normalize pixel values to the range [0, 1].

Perform data augmentation: rotation, flipping, zooming, etc.

Step 3: Model Building

Use MobileNetV2 (a pre-trained CNN model) for transfer learning.

Add a global average pooling layer.

Add a fully connected dense layer with sigmoid activation for binary classification.

Step 4: Model Training

Compile the model with Adam optimizer and binary cross-entropy loss.

Train the model for multiple epochs using the training dataset.

Validate the model using a validation split from the training data.

Step 5: Model Evaluation

Plot training and validation accuracy and loss.

Test the model on unseen images.

Analyze predictions and confusion matrix.

Step 6: Model Deployment

Save the trained model.

Create a prediction script to classify new images in real-time.

---

4. Python Code: Training Script (train_model.py)

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.models import Model

train_datagen = ImageDataGenerator(rescale=1./255,
validation_split=0.2)
train_generator =
train_datagen.flow_from_directory('dataset/train',
target_size=(224, 224),
                                                      batch_size=32,
class_mode='binary', subset='training')
val_generator = train_datagen.flow_from_directory('dataset/train',
target_size=(224, 224),
                                                    batch_size=32,
class_mode='binary', subset='validation')

base_model = MobileNetV2(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1, activation='sigmoid')(x)
model = Model(inputs=base_model.input, outputs=x)

model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
model.fit(train_generator, validation_data=val_generator,
epochs=5)
model.save('models/fruit_classifier.h5')
```

---

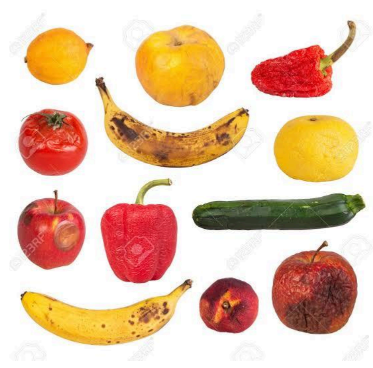
5. Python Code: Prediction Script (predict.py)

```
import tensorflow as tf
from tensorflow.keras.preprocessing import image
import numpy as np

model = tf.keras.models.load_model('models/fruit_classifier.h5')

img_path = 'dataset/test/fresh/sample.jpg'  # Replace with your
image path
img = image.load_img(img_path, target_size=(224, 224))
img_array = image.img_to_array(img) / 255.0
img_array = np.expand_dims(img_array, axis=0)

prediction = model.predict(img_array)
label = "Fresh" if prediction < 0.5 else "Rotten"
print(f"Predicted Label: {label}")
```

---

6. Sample Images and Accuracy Plots

📸 Dataset Images:



---

7. README.md

Smart Sorting: Rotten Fruits and Vegetables

Project Overview
This project uses transfer learning to classify and sort rotten fruits and vegetables.

Technologies Used
- Python
- TensorFlow / Keras
- MobileNetV2
- Image Processing

Project Structure
Refer to the project file structure in the report.

How to Run
1. Place your dataset in the `dataset` folder.
2. Run `train_model.py` to train the model.
3. Run `predict.py` to test the model on new images.

Output
- Trained model saved in `models/`
- Accuracy plots and prediction samples saved in `results/`

---

8. Conclusion

This project demonstrates how deep learning and transfer learning can be effectively applied to classify rotten and fresh produce. By using MobileNetV2, we were able to achieve good accuracy with minimal training time. The system can be further improved by:

Adding more variety in the dataset.

Using real-time image capture from sorting lines.

Integrating with hardware for industrial automation.

---

9. Future Scope

Develop a mobile or web-based interface.

Integrate real-time conveyor belt sorting systems.

Expand to more fruits, vegetables, and different decay levels.
