Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables

A Project Report Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor of Technology

Submitted by: Illuru Karthik [22AT13136]

Under the Esteemed Guidance of: Dr. T.Tirupal sir

College: G.Pullaiah college of Engineering and Technology.

Department: B.Tech (Artificial Intelligence & Machine Learning) [Academic Year, e.g., 2024-2025]

Abstract

This project introduces "Smart Sorting," a pioneering AI-driven system meticulously engineered for the binary classification of fruits and vegetables into 'fresh' or 'rotten' categories. At its core, the system harnesses the formidable power of deep learning, specifically **transfer learning**, to address the pervasive issue of food spoilage detection. By leveraging the pre-trained knowledge embedded within the **MobileNetV2** convolutional neural network architecture, the system efficiently adapts to a custom-curated dataset of produce images, thereby mitigating the need for vast quantities of training data from scratch.

The technical implementation encompasses a robust machine learning backend, where the fine-tuned MobileNetV2 model performs image inference with high accuracy. This intelligence is seamlessly integrated with a user-friendly **web interface developed using Flask**, a lightweight Python micro-framework. The interface allows users to intuitively upload images, upon which the system processes the input and provides an immediate, clear prediction regarding the produce's freshness, complemented by a confidence score.

"Smart Sorting" transcends a mere academic exercise; it represents a tangible solution with profound implications for real-world applications. Its potential to significantly enhance operational efficiency and reduce waste is considerable across various sectors, including large-scale agricultural sorting facilities, modern supermarket inventory management, and even

integrated smart home environments, promoting sustainable practices and economic benefits by minimizing post-harvest losses.

Introduction

The global food supply chain faces persistent challenges, among the most critical being post-harvest losses due to spoilage. Manual inspection and sorting of fruits and vegetables, while traditional, are inherently labor-intensive, highly susceptible to human error, and notoriously inefficient. These limitations contribute significantly to food waste, economic setbacks for producers and retailers, and diminished consumer satisfaction. Estimates suggest that a substantial portion of agricultural produce perishes before reaching consumers, a problem exacerbated by inadequate sorting mechanisms.

This project, "Smart Sorting," proposes an innovative and automated solution to these challenges by embracing the advancements in Artificial Intelligence (AI), particularly in the domains of machine learning and computer vision. Our system aims to revolutionize the process of identifying spoiled produce, transforming it from a subjective, manual task into an objective, automated, and highly efficient operation.

The project integrates a sophisticated **machine learning model** capable of discerning subtle visual cues indicative of spoilage, a robust **computer vision pipeline** for image preprocessing and feature extraction, and a practical **web development framework (Flask)** to deliver an accessible and end-to-end classification platform. By automating the quality assessment of fruits and vegetables, "Smart Sorting" endeavors to significantly reduce waste, optimize supply chain logistics, and contribute to more sustainable food systems. This document details the

methodology, implementation, evaluation, and potential impact of this AI-powered sorting solution.

Problem Statement:

The central problem addressed by this project is the inefficiencies and inaccuracies inherent in traditional, manual methods of identifying rotten fruits and vegetables. Current practices are characterized by:

- **High Labor Costs**: Sorting produce manually requires significant human resources, leading to high operational expenses for farms, warehouses, and retailers
- **Inconsistent Accuracy**: Human fatigue, subjective judgment, and varying levels of expertise lead to inconsistencies in identifying spoilage, often resulting in good produce being discarded prematurely or rotten produce entering the supply chain.
- **Time Consumption**: Manual sorting is a slow process, creating bottlenecks in high-volume environments and limiting throughput, especially for perishable goods with limited shelf life
- **Significant Food Waste**: The combined effect of slow processing and inaccurate sorting leads to substantial amounts of edible produce being discarded, contributing to global food waste issues and economic losses
- **Lack of Real-Time Feedback**: Existing methods do not provide immediate, objective assessments of produce quality, hindering proactive decision-making in logistics and inventory management

Therefore, there is an urgent need to develop an **AI-based solution** that can reliably and efficiently classify images of fruits and vegetables as either "fresh" or "rotten." This solution must provide a **user-friendly and intuitive interface** that allows for quick image submission and delivers **real-time results**, thereby overcoming the limitations of manual inspection and offering a scalable, objective, and automated alternative for quality control in the produce industry

Objective:

The overarching objective of the "Smart Sorting" project is to develop and validate an intelligent system for automated produce classification. This includes several specific, measurable, achievable, relevant, and time-bound (SMART) objectives:

- **To train a robust and accurate image classification model:** The primary aim is to develop a deep learning model capable of distinguishing between fresh and rotten produce. This will be achieved through the effective application of **transfer learning**, specifically by fine-tuning the pre-trained MobileNetV2 architecture on a focused dataset of apple images. The goal is to maximize prediction accuracy and generalization capability while minimizing the training data requirement.
- **To develop a web-based user interface (UI) for image submission:** Create an intuitive and accessible frontend using HTML and CSS, backed by a Flask server. This interface will allow users to seamlessly upload digital images of fruits or vegetables for immediate analysis, without requiring specialized software or technical expertise.
- **To integrate the trained machine learning model with the web UI for real-time inference:** Establish a seamless communication pipeline between the Flask backend and the saved .h5 model. This integration will enable the system to process uploaded images, perform inference using the trained model, and display the classification result (fresh/rotten) along with a confidence score to the user in near real-time
- **To demonstrate practical applications and scalability:** Showcase the versatility and potential impact of the "Smart Sorting" system through compelling use-case scenarios relevant to various industries (e.g., supermarkets, food processing plants) and domestic settings (e.g., smart homes), highlighting its ability to reduce food waste and improve efficiency.
- **To achieve a minimum acceptable classification accuracy:** Aim for a validation accuracy of at least 85-90% on the test dataset for the chosen produce category (apples), demonstrating the model's reliability in practical scenarios

By achieving these objectives, "Smart Sorting" will provide a tangible, AI-powered solution for automated produce quality assessment, addressing critical industry needs and contributing to sustainability efforts.

Technologies Used:

The "Smart Sorting" project integrates a strategic selection of modern technologies across its frontend, backend, and machine learning components. For the **frontend**, standard web languages **HTML** and **CSS** were utilized. HTML structured the web pages, including image upload forms and result display areas, while CSS styled the interface for a clean and user-friendly experience, ensuring responsive design and visual appeal. The project's **backend** is built with **Python**, a versatile programming language that orchestrates the application logic. **Flask**, a lightweight Python micro web framework, was chosen for its simplicity and efficiency in handling web requests, managing file uploads, and rendering dynamic templates.

At the core of the system lies its **machine learning** capabilities, powered primarily by **TensorFlow** and **Keras**. TensorFlow, a robust open-source framework, provides the foundation for building and deploying ML models. Keras, running on TensorFlow, offered a high-level API for rapid model development and training. The specific model architecture employed is **MobileNetV2**, a highly efficient pre-trained convolutional neural network. MobileNetV2's pre-trained weights from ImageNet were leveraged through **transfer learning**, significantly reducing the need for extensive custom data and training time, while providing powerful feature extraction for classifying fresh versus rotten produce. Supporting these core technologies, **OpenCV** was used for essential image processing tasks like resizing and normalization. **Google Colab** provided the necessary GPU resources for model

training, and **Git & GitHub** managed version control and collaborative development of the codebase, ensuring a streamlined and organized project workflow.

Dataset Used:

The efficacy of the "Smart Sorting" model heavily relies on the quality and characteristics of its training data. The dataset utilized for this project was sourced from **Kaggle**, a well-known platform for public datasets. This dataset was specifically curated for a binary classification task, featuring images categorized into "Fresh Apples" and "Rotten Apples."

In total, approximately **200 images** were used. This modest size for a deep learning project was effectively managed through the strategic application of transfer learning. The dataset was divided into distinct subsets for training and testing: **150 images** were allocated to the **training set**, comprising an equal distribution of 75 'Fresh Apples' and 75 'Rotten Apples' to ensure balanced learning. The remaining **50 images** formed the **testing set**, with 25 images from each class, specifically reserved for unbiased evaluation of the trained model's performance on unseen data. The images were available in common formats such as JPG, PNG, and WebP. Crucial preprocessing steps were applied to all images before model input; every image was uniformly **resized to 224×224 pixels** to meet model input requirements, and their pixel values were **normalized** to optimize the

learning process, ensuring consistent data representation for the neural network. This careful preparation of the dataset was fundamental to achieving the observed high accuracy despite its relatively small scale.

Model Architecture:

The "Smart Sorting" project employs a sophisticated deep learning architecture centered on **transfer learning**, specifically utilizing the pre-trained **MobileNetV2** convolutional neural network as its foundational backbone. Transfer learning is a powerful technique that leverages a model pre-trained on a vast dataset (like ImageNet) to perform well on a new, related task with limited custom data.

The **Base Model**, MobileNetV2, was chosen for its optimal balance of high accuracy and computational efficiency, making it suitable for practical applications. Its layers were **frozen** during our training phase, preserving the rich, generic visual features learned from ImageNet. This approach allowed the model to effectively transfer its deep understanding of image patterns to our specific task without requiring extensive retraining of its core layers. On top of this frozen base, a **Custom Classification Head** was appended. This head consists of a `GlobalAveragePooling2D` layer to efficiently reduce feature map dimensions, followed by a `Dense layer` with 128 neurons and **ReLU activation** for learning task-specific patterns. A `Dropout` layer (rate 0.3) was included to prevent overfitting by randomly deactivating neurons during training. The final layer is an **Output `Dense` layer** with a single neuron and **Sigmoid activation**, providing a probability score between 0 and 1 for binary classification (fresh/rotten).

The model was compiled with **Binary Crossentropy** as the loss function, which is ideal for binary classification problems, and the **Adam optimizer** for efficient weight updates. It was trained for **5 epochs**. Through this architecture and training configuration, the model achieved a commendable **validation accuracy of approximately 90%**, demonstrating its strong capability in distinguishing between fresh and rotten produce efficiently.

Training process:

The training process for the "Smart Sorting" model was systematically executed to ensure optimal learning and robust performance from our dataset. The initial phase involved **dataset upload and environment setup** within **Google Colab**. This cloud-based platform provided crucial access to GPUs, significantly accelerating the deep learning model training. All necessary libraries, including TensorFlow and Keras, were installed and configured within the Colab environment.

Following setup, **Image Preprocessing and Data Augmentation** were performed using Keras's **`ImageDataGenerator`**. This tool efficiently handled on-the-fly transformations. Crucially, `ImageDataGenerator` performed **data augmentation** (e.g., slight rotations, shifts, flips) on the training images. This technique artificially increased the effective size and diversity of our relatively small dataset, thereby significantly reducing overfitting and improving the model's ability to generalize to unseen images. Images were consistently resized to 224×224 pixels and normalized.

Next, the **Model Loading and Compilation** phase commenced. The pre-trained **MobileNetV2** base model was loaded with its top classification layers excluded, and its base layers were explicitly **frozen** to preserve their learned features from ImageNet. Our custom classification head, comprising `GlobalAveragePooling2D`, `Dense`, `Dropout`, and an `Output Dense` layer, was then added on top. The complete model was then **compiled**, defining the **Adam optimizer**, **Binary Crossentropy** as the loss function, and `accuracy` as the primary metric for monitoring performance.

The actual **Model Training** was conducted using the `model.fit()` method, supplying it with both the augmented training data and validation

data. This allowed for real-time monitoring of performance on unseen examples. The model was trained for **5 epochs**, during which it iteratively adjusted the weights of its unfrozen layers. Throughout training, **Accuracy and Loss Plots were generated**, providing visual insights into the model's learning progression and helping to diagnose any overfitting or underfitting. Upon completion, the final trained model, including its architecture and weights, was saved as `smart_sorting_model.h5`, ready for deployment and inference. This structured process ensured the development of an accurate and robust classification system.

Model Evalution:

Model evaluation is a critical step that objectively assesses how well the trained "Smart Sorting" model performs on unseen data, confirming its readiness for practical application. The primary metric observed was the **Validation Accuracy**, which reached approximately **90%**. This high accuracy signifies that the model correctly classified nearly 9 out of every 10 images from the unseen validation dataset. This strong performance on validation data is a robust indicator of the model's ability to **generalize** effectively, meaning it can reliably predict the freshness of new, real-world fruit images not encountered during its training.

Further insights into the model's performance were gained through **Confusion Matrix Analysis**. While specific numerical values of the matrix are not detailed here, the assessment indicated a **high number of True Positives**, meaning the model was highly successful at correctly identifying rotten apples when they were indeed rotten. Simultaneously, there was a **low number of False Positives**, indicating that the model rarely misclassified fresh apples as rotten, which is crucial for minimizing unnecessary waste of good produce.

In conclusion, the model demonstrated exceptional performance, particularly considering the relatively small size of our custom dataset. This success is predominantly attributed to the effective application of **transfer learning**. By leveraging the deep, pre-learned features from MobileNetV2, the model efficiently adapted to the nuanced distinctions between fresh and rotten

produce, minimizing the need for extensive, domain-specific data and confirming its reliability for practical deployment in food spoilage detection.

Web Interface(Flask) :

To render the "Smart Sorting" machine learning model accessible and user-friendly, a practical web interface was developed using the **Flask** microframework. This interface acts as the primary point of interaction, allowing users to effortlessly upload images and receive real-time classification results without requiring any specialized technical expertise.

The core of the web application is a **simple Flask server**, chosen for its lightweight nature and flexibility, which efficiently handles HTTP requests, manages file uploads, and renders dynamic HTML templates. The interface facilitates **image upload functionality** via a straightforward HTML form, enabling users to select an image file from their device. Upon submission, the Flask backend processes the uploaded image. This involves crucial **image preprocessing**, such as resizing the image to 224×224 pixels and normalizing its pixel values, to match the input specifications of the loaded `smart_sorting_model.h5`. The preprocessed image is then fed into the model for **prediction**, which returns a probability score. This score is then interpreted into a clear "Fresh" or "Rotten" classification based on a predefined threshold.

The results are dynamically displayed on the web page using **Jinja2 templates**, which Flask utilizes to render dynamic HTML content. This ensures that the **prediction display** is

immediate, showing the "Fresh" or "Rotten" status along with a **confidence score**, providing users with valuable context about the model's certainty. The web interface transforms the complex underlying machine learning capabilities into an intuitive, accessible, and interactive tool, making "Smart Sorting" a complete end-to-end solution for practical use.

Testing:

Thorough testing was conducted to rigorously validate the functionality, accuracy, and overall user experience of the "Smart Sorting" system. This phase was crucial for confirming the seamless integration between the web interface and the machine learning backend, as well as verifying the accuracy of predictions across various test cases.

The testing methodology began with deploying the Flask web application on a local development server. A diverse set of test images, distinct from the training and validation sets, was prepared, including both unambiguously "fresh" and "rotten" apples. A specific and illustrative test case involved uploading an image named `testapple.webp`. The testing workflow confirmed that upon upload, the Flask server successfully received and preprocessed the image (resizing and normalization) according to the model's input requirements. Subsequently, the preprocessed image data was correctly passed to the loaded `smart_sorting_model.h5`, and the model performed its classification, generating a prediction. Critically, the prediction (e.g., "Prediction: Rotten, Confidence: 97%") was then accurately and promptly displayed on the web user interface, consistent with the `testapple.webp`'s known ground truth.

Beyond this specific example, multiple test cases were systematically tried, encompassing various fresh and rotten apple images to assess the system's robustness across different scenarios. This comprehensive testing validated the smooth, end-to-end functionality of the "Smart Sorting" system, demonstrating that it is a reliable and accurate application ready for practical demonstration.

Use Case Scenarios:

The "Smart Sorting" system, by automating the identification of rotten fruits and vegetables, offers substantial value across numerous industries and even in domestic environments. Its ability to provide rapid, objective, and consistent quality assessments translates directly into reduced waste, enhanced operational efficiency, and improved quality control throughout the food supply chain.

In **Supermarkets and Retail Stores**, "Smart Sorting" can revolutionize quality control at receiving docks, allowing for immediate identification and segregation of spoiled incoming produce, preventing contamination and reducing initial losses. During shelf stocking, it can help employees quickly assess freshness, ensuring only high-quality items are displayed, which enhances customer satisfaction and minimizes in-store waste. For **Smart Refrigerators and Home Appliances**, integrating this technology could enable automatic spoilage detection. Cameras inside smart refrigerators could periodically scan produce and notify users of items nearing spoilage, thereby significantly reducing household food waste and promoting sustainable consumption.

In **Food Processing and Packaging Plants**, "Smart Sorting" can be deployed on conveyor belts. High-speed cameras and the

system's analysis can trigger automated mechanisms to quickly divert rotten items, drastically increasing throughput, reducing manual labor, and ensuring only high-quality raw materials proceed for processing or packaging. Similarly, **Wholesale and Retail Warehouses** can utilize the system for improved inventory management and rotation, rapidly assessing quality during storage to prioritize dispatch of items nearing spoilage, which minimizes losses from extended storage. Ultimately, "Smart Sorting" offers a versatile solution with transformative potential to create more efficient, sustainable, and less wasteful food supply chains from farm to fork.

Challenges Faced:

The development of the "Smart Sorting" project, while successful, navigated several technical and practical challenges. Addressing these issues required careful problem-solving and strategic implementation choices.

One significant hurdle was the **difficulty in finding clean, accurately labeled datasets specifically for distinguishing fresh from rotten produce**. While general image datasets are common, obtaining high-quality images covering various stages of spoilage for different produce types proved challenging due to inconsistencies in labeling or limited variations. This was primarily mitigated by focusing on a specific produce (apples) initially and crucially, by leveraging **transfer learning** and **data augmentation** techniques, which lessened the reliance on a massive custom dataset by expanding the effective training data.

Another challenge was ensuring **training performance on limited data**. Deep learning models typically require vast amounts of data to prevent overfitting. With our relatively small dataset, the risk of the model simply memorizing training examples rather than learning generalized features was high. This was addressed by strategically **freezing the base MobileNetV2 layers**, only training the smaller, newly added classification head. The inclusion of a **Dropout layer (0.3)** was also vital as a regularization technique to prevent co-adaptation of features and improve the model's ability to generalize to unseen data.

Finally, the **integration of the machine learning model with Flask** presented its own set of challenges. This involved correctly loading the

trained `.h5` model within the web application, handling image file uploads, processing them to the model's input format, and then seamlessly returning the prediction to the user interface. Ensuring **compatibility with the .h5 format** across different environments also required careful attention. These integration complexities were overcome by utilizing Keras's `load_model` function, Flask's robust request handling, and OpenCV for precise image preprocessing within the backend, ensuring a smooth end-to-end workflow from image upload to prediction display. Overcoming these challenges was integral to the successful deployment of a functional and accurate "Smart Sorting" system.

Conclusion:

The "Smart Sorting" project successfully demonstrates the immense potential of integrating deep learning and computer vision to address a pervasive real-world challenge: the efficient and accurate identification of rotten fruits and vegetables. By developing an AI-powered system that classifies produce as 'fresh' or 'rotten', this project has offered a compelling solution to mitigate food waste and enhance quality control across various sectors.

The core of our success lies in the judicious application of **transfer learning**, specifically leveraging the pre-trained **MobileNetV2** architecture. This strategic choice allowed us to achieve a remarkable **validation accuracy of approximately 90%** on our custom dataset of apple images, despite its relatively limited size. This achievement underscores the efficiency of transfer learning in rapidly adapting powerful pre-existing neural network knowledge to new, specific domains, thereby reducing both data dependency and computational training costs.

Furthermore, the development of an intuitive **web interface using Flask** democratizes access to this advanced AI capability. Users can effortlessly upload images and receive immediate, clear predictions, transforming a complex machine learning model into a practical and user-friendly tool. The comprehensive testing validated the seamless integration of the frontend, backend, and the machine learning inference pipeline, confirming the system's robustness and reliability. The identified **use case scenarios**—ranging from supermarkets and food processing plants to

smart home environments—underscore the broad applicability and significant economic and environmental benefits of "Smart Sorting." By automating quality assessment, the system promises to reduce manual labor, improve sorting accuracy, decrease post-harvest losses, and ultimately contribute to more sustainable and efficient food supply chains. In conclusion, "Smart Sorting" stands as a testament to how cutting-edge AI methodologies can be harnessed to solve tangible problems, offering an efficient, cost-effective, and scalable solution for food spoilage detection that is well-positioned for future expansion and real-world deployment.

Future Work:

While the "Smart Sorting" project successfully established a functional and accurate system for classifying fresh and rotten apples, the field of AI and its applications are continuously evolving, presenting several clear avenues for further development and enhancement.

The most critical next step is to **expand the dataset with more fruit and vegetable categories**. The current model is specialized for apples, but broadening its scope to include a diverse array of produce (e.g., bananas, tomatoes, leafy greens) would significantly enhance its versatility and real-world applicability, leading to a general-purpose produce sorter. Another key enhancement involves adding **object detection capabilities to localize rotten spots**. Instead of just classifying an entire image as rotten, future work could implement advanced computer vision techniques like YOLO or Faster R-CNN to draw bounding boxes around specific areas of spoilage or defects, providing more granular and actionable insights for quality control.

Further improvements include enhancing the **frontend with a better design and mobile support**. Redesigning the web

interface using modern frameworks and ensuring full mobile responsiveness would create a more intuitive, visually appealing, and accessible user experience across various devices. For broader reach and continuous operation, **deployment on cloud platforms** like Render or Replit is essential. This would make the "Smart Sorting" system publicly accessible 24/7, enabling wider testing, demonstrations, and potential integration with other systems. Ultimately, the system could be transformed into a **mobile app or integrated with IoT devices**. Developing native Android/iOS apps or connecting with smart cameras in refrigerators or on industrial conveyor belts would enable fully autonomous, real-time monitoring and sorting, moving towards pervasive, smart solutions for food quality management. These advancements will evolve "Smart Sorting" into a more comprehensive, powerful, and widely applicable tool for combating food spoilage and promoting sustainability in the food industry.