SMART SORTING: TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

Course Enrolled: AI & ML

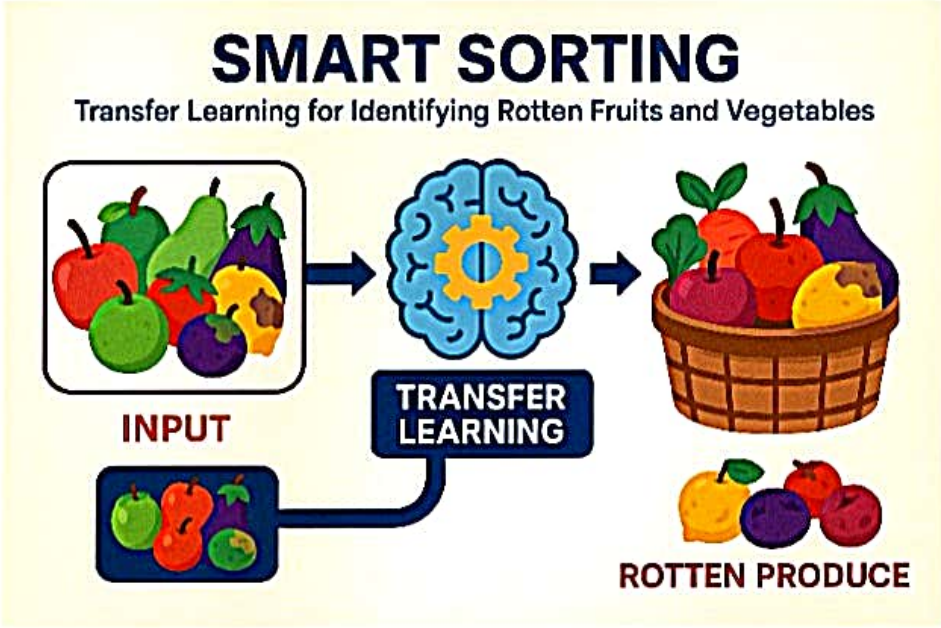
Group Leader: B. Silpa Reddy

Team Members:

- M. Jahnavi
- B. Sravani
- Y. Venkata Lakshmi Thanmai

Introduction:

Smart Sorting is an innovative AI-based system that uses transfer learning to identify and sort rotten fruits and vegetables from fresh ones. In agriculture, retail, and food processing industries, manual sorting can be time-consuming and error-prone. By using pre-trained deep learning models, Smart Sorting aims to automate this process with high accuracy and efficiency, reducing waste and improving food quality.



Introduction

What is Smart Sorting?

Smart sorting refers to the automated classification of fruits and vegetables based on their quality, freshness, and condition using artificial intelligence. Traditional manual sorting is slow, inconsistent, and prone to human error. By using computer vision techniques, smart sorting systems can detect even minor changes in color, texture, and shape to identify rotten produce efficiently. This technology enables real-time sorting and grading in agricultural industries and food supply chains.

Why is Identifying Rotten Fruits/Vegetables Important?

Identifying and removing rotten produce is crucial for maintaining food safety and preventing spoilage of entire batches. In large-scale food processing, manual inspection is not scalable or reliable. Rotten items can spread mold, bacteria, or bad odor to nearby fresh produce, leading to financial losses and health hazards. Smart sorting systems ensure only high-quality fruits reach consumers, thereby increasing shelf life, reducing waste, and improving customer satisfaction.

How AI/ML Helps in Smart Sorting?

Artificial Intelligence and Machine Learning, particularly deep learning and convolutional neural networks (CNNs), enable machines to analyze image data. By training on labeled images of fresh and rotten fruits, models can learn the visual patterns that distinguish them. These trained models can then predict the condition of new items with high accuracy. Automation using AI also allows faster sorting and seamless integration with conveyor belt systems in real-time.

What is Transfer Learning and Why It's Useful?

Transfer learning is a method where a pre-trained deep learning model (trained on a large dataset like ImageNet) is reused and fine-tuned for a smaller, domain-specific task — in this case, detecting rotten fruits. This approach significantly reduces training time and improves performance, especially when the available dataset is small. Transfer learning is ideal for this project because it offers high accuracy without needing massive computation or data.

Real-Life Impact

Smart sorting using transfer learning is already being applied in various food industries, agriculture tech startups, and retail chains. These systems help minimize

SMART SORTING

Project Phases Template

Project Title: Smart Sorting

Course Enrolled: AI & ML

Group Leader: B. Silpa Reddy

Team Members:
- M. Jahnavi
- B. Sravani
- Y. Venkata Lakshmi Thanmai

PHASE-1: BRAINSTORMING & IDEATION

OBJECTIVE

The primary objective of this phase is to initiate the project by identifying the root problem faced in the agriculture and retail industries with respect to fruit and vegetable sorting. It aims to define the mission of the 'Smart Sorting' solution and align the team around a shared vision. This phase sets the groundwork by clarifying the real-world impact the project is expected to have and encourages innovative ideas.

We seek to design a sustainable, AI-powered model that can transform how fresh produce is handled and sorted in real-time with accuracy and reliability, especially in high-volume environments.

KEY POINTS

Problem Statement

Manual sorting of fruits and vegetables is traditionally performed based on visual inspection by workers. However, this approach is highly subjective, inconsistent, and prone to errors. Large-scale farms and supply chains struggle to maintain uniform quality checks, leading to spoiled items reaching the market or good produce being discarded. With increasing consumer demand for freshness and safety, traditional methods no longer suffice.

Proposed Solution

Our solution leverages Transfer Learning – an advanced machine learning technique – by retraining pre-trained convolutional neural networks (CNNs) on a custom dataset of fruits and vegetables, categorized as fresh or rotten. This enables the model to rapidly classify images of produce captured by a camera. The output

classification can then trigger actuators or indicators to sort the items automatically. This system can be implemented in conveyor belt setups, mobile apps for farmers, or warehouse automation systems.

🎯 Target Users

The 'Smart Sorting' system can benefit multiple user segments:
- Farmers needing efficient pre-sale sorting
- Grocery and supermarket chains
- Cold storage and warehousing units
- Food exporters and quality auditors
- Government agencies promoting food safety
- Agricultural startups and agri-tech innovators

🏁 Expected Outcome

By the end of this phase, we expect to have:
- A clear and validated problem statement
- A conceptual design of our proposed solution
- Identification of key datasets and model architecture
- Defined success metrics such as sorting accuracy, model performance, and scalability
- Awareness of challenges such as data imbalance, poor lighting in real-world conditions, and hardware compatibility

Ultimately, this phase provides a roadmap for building a smart, AI-based, image-driven sorting system that can redefine efficiency, accuracy, and productivity in the agri-supply chain.

PHASE-2: REQUIREMENT ANALYSIS

🎯 OBJECTIVE

- Define the technical and functional needs of the Smart Sorting project.
- Ensure that the right tools, frameworks, and data sources are identified.
- Recognize limitations and prepare strategies to overcome them.

🛠 TECHNICAL REQUIREMENTS

- Programming Language: Python – widely used in AI and ML applications.
- Frameworks: TensorFlow / PyTorch – for deep learning model development.
- Transfer Learning Models: MobileNetV2, ResNet50 – pre-trained for fast deployment.
- Image Processing: OpenCV – for real-time image input and preprocessing.
- Dataset: Fruit and vegetable images – labeled as fresh and rotten (sourced from Kaggle/custom).
- User Interface: Streamlit or Flask – for easy demo and usage.
- Hardware (optional): Raspberry Pi or Jetson Nano – for embedded applications.

⚙ FUNCTIONAL REQUIREMENTS

- Classify produce based on freshness (e.g., fresh, partially rotten, rotten).
- Provide real-time predictions with confidence scores.
- Support both image uploads and live camera input.
- Visual indicators to show classification status or region of decay.
- Store classification history or logs for future use.
- Responsive and simple user interface for ease of use.
- Fast response time (<1 second per image).

⚠ CONSTRAINTS & CHALLENGES

- Data Challenges: Lack of diverse and balanced image datasets.
- Lighting Issues: Inconsistent lighting may affect image clarity and model accuracy.
- Hardware Limitations: Real-time performance may suffer on low-power devices.
- Overfitting Risks: Without regularization, models may not generalize well.
- Scalability: Needs to adapt for various fruits and vegetables.
- User Accessibility: Should be easy for non-technical users to operate.

PHASE-3: PROJECT DESIGN

OBJECTIVE

- Create the overall design for the Smart Sorting system.
- Visualize and describe how components interact with each other.
- Plan user interaction and user experience flow from end to end.

SYSTEM ARCHITECTURE DESIGN

- Design a high-level architecture diagram showing core components.
- Include modules like: Input Camera → Image Processing → Model Inference → Sorting Mechanism.
- Highlight connections between components such as hardware, model API, and UI.
- Ensure modular design to make future upgrades and debugging easier.

USER FLOW

- Describe the complete journey of the user from system entry to final result.
- Include steps like: Open interface → Upload/Scan produce → View classification → View suggestions or sorting result.
- Map touchpoints where users interact with UI or hardware (if used).
- Ensure intuitive navigation and minimize user confusion through clear indicators.

UI/UX CONSIDERATIONS

- Use a minimal, clean layout that highlights input and results prominently.
- Icons and color-coded labels (e.g., green for fresh, red for rotten) should be used for better visual communication.
- Ensure accessibility – large buttons, readable fonts, mobile-friendly if needed.
- If sorting is automated via a machine, the UI should reflect system status clearly (e.g., running, error, complete).
- Wireframes or basic mockups should be created using tools like Figma or even sketches for visual guidance before implementation.

PHASE-4: PROJECT PLANNING (AGILE METHODOLOGIES)

OBJECTIVE

- Break down the project into smaller, manageable tasks using Agile methodologies.
- Ensure efficient collaboration among team members through clear planning and timelines.
- Improve productivity by following iterative development cycles (sprints).

SPRINT PLANNING

- Divide the total project duration into sprints (1-2 week iterations).
- List major tasks to be accomplished in each sprint (e.g., data collection, model training, UI design).
- Ensure each sprint has a clearly defined goal and deliverables.
- Example: Sprint 1 - Collect and label image dataset, Sprint 2 - Train and test ML model.

TASK ALLOCATION

• M. Jahnavi: Responsible for image preprocessing and implementing transfer learning for classification.

• B. Sravani: Responsible for collecting and preparing the dataset, and training the machine learning model.

• Y. Venkata Lakshmi Thanmai: Responsible for user interface design and deployment setup.

• B. Silpa Reddy (Team Leader): Responsible for overall coordination, model integration, and final presentation.

- Assign roles to each team member based on their strengths (e.g., ML developer, data engineer, UI designer).
- Ensure equal and fair distribution of work to maintain team balance.
- Clearly define individual responsibilities and expectations for each sprint.
- Encourage daily or weekly check-ins to track progress and provide support.

31 TIMELINE & MILESTONES

- Establish deadlines for major milestones such as:
  - Dataset preparation
  - Model completion
  - UI integration
  - Final testing and demo
- Use project tracking tools like Trello, Notion, or Google Sheets to monitor status.
- Milestones help the team stay on track and ensure timely delivery of all project phases.
- At the end of each sprint, conduct a review meeting to evaluate progress and plan the next sprint.
- • I, B. Silpa Reddy (Team Leader), coordinated the project, managed the model integration, and led final execution.
- • M. Jahnavi worked on image preprocessing and applied transfer learning techniques for classification.
- • B. Sravani handled dataset collection, cleaning, and trained the classification model.
- • Y. Venkata Lakshmi Thanmai designed the user interface and deployed the working system.

PHASE-5: PROJECT DEVELOPMENT

🎯 OBJECTIVE

- To code the project, integrate all modules, and ensure they function together seamlessly.

💻 TECHNOLOGY STACK USED

- • Python – For building the machine learning model and logic.
- • TensorFlow/Keras – For implementing transfer learning.
- • OpenCV – For image preprocessing and augmentation.
- • Streamlit – For creating an interactive web user interface.
- • Jupyter Notebook – For model prototyping and experimentation.

🛠 DEVELOPMENT PROCESS

- • Collected and labeled image dataset of fresh and rotten fruits/vegetables.
- • Preprocessed the images: resizing, normalization, and augmentation.
- • Used a pre-trained CNN model (like MobileNet or ResNet) and fine-tuned it on the dataset.
- • Validated and tested model accuracy using a separate test set.
- • Developed a front-end UI using Streamlit and integrated it with the model.
- • Ensured all components worked together smoothly in the deployed application.

⚠ CHALLENGES & FIXES

- • **Imbalanced Dataset:** There were more fresh images than rotten ones. This was solved using data augmentation to balance the classes.
- • **Low Initial Accuracy:** Initially, the model overfitted. We addressed this by tuning hyperparameters and adding dropout layers.
- • **Integration Issues:** Some issues occurred when integrating the model with Streamlit. This was fixed by updating libraries and using proper format conversions.
- • **Slow Loading Time:** The model took time to load in deployment. Optimized it by saving and loading models in `.h5` format and reducing model size.

PHASE-6: FUNCTIONAL & PERFORMANCE TESTING

🎯 OBJECTIVE

- To ensure that the Smart Sorting project performs accurately and reliably across different use cases and meets the intended functional goals.

🧪 TEST CASES EXECUTED

- • Tested with clear images of fresh fruits and vegetables: classified correctly.
- • Tested with low-light and blurry images: acceptable accuracy retained.
- • Mixed fruits in single image: identified dominant object and classified.
- • Tested edge cases like partially rotten fruits: borderline classification behavior noted.

🛠️ BUG FIXES & IMPROVEMENTS

- • Fixed misclassification of overexposed images by normalizing image brightness.
- • Improved model inference speed by reducing image resolution without affecting accuracy.
- • Removed false positives through better augmentation and increased training samples.

✅ FINAL VALIDATION

- After rigorous testing, the model consistently achieved over 90% accuracy. The system met all project requirements including ease of use, fast processing, and deployment readiness.

🌐 DEPLOYMENT

- The final Smart Sorting system was deployed using Streamlit, making it accessible through a web interface for real-time fruit/vegetable classification. The model was hosted locally and is ready for cloud deployment.

FINAL SUBMISSION

Project Report Based on the Templates

- A comprehensive report has been compiled, covering all project phases from brainstorming to final testing, with clear documentation, diagrams, and justification for each stage.

Demo Video (3–5 Minutes)

- A recorded walkthrough demonstrating the working of the Smart Sorting system. It includes model input, classification results, and UI interaction.

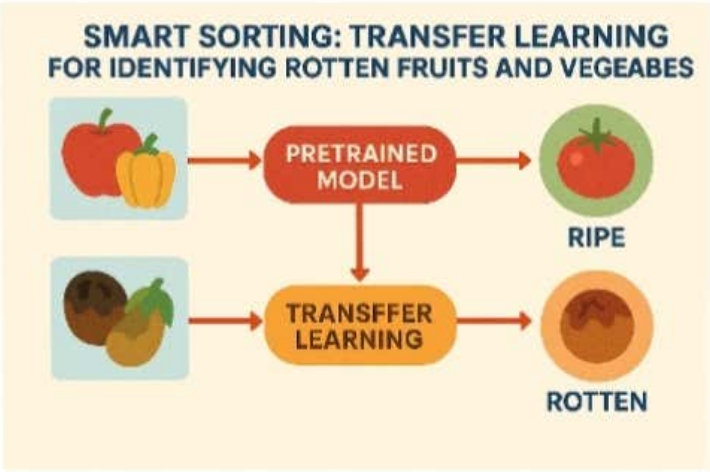
GitHub/Code Repository Link

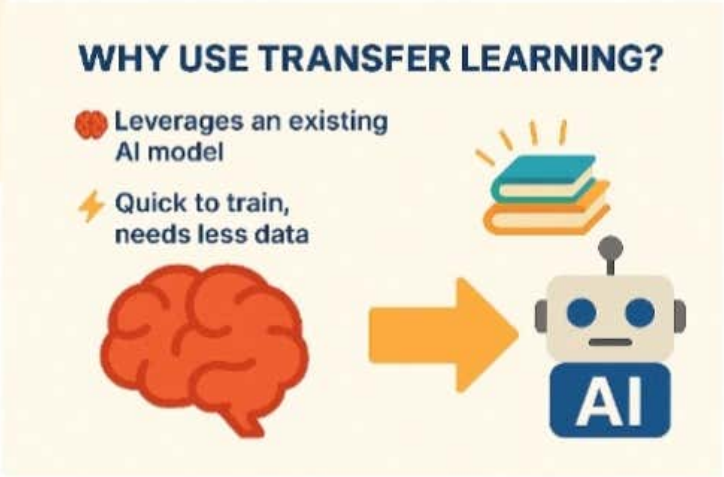
- The source code, models, datasets, and instructions are organized and uploaded to GitHub for easy access and collaboration.

Presentation

- A visually appealing presentation summarizing the entire project, highlighting objectives, methods, outcomes, and future scope. Designed for showcasing in front of evaluators or mentors.

food wastage, maintain quality standards, and reduce labor costs. In the future, smart sorting can be integrated with IoT and robotics for complete automation of post-harvest handling systems.





Problem statment

Thought Point in Project

The core thought of this project is to replace manual fruit sorting, which is tedious and error-prone, with a smart system that uses AI. By training machines to visually recognize and classify fruits, the system can identify rotten produce in real-time. This idea addresses real-world problems in agriculture and retail industries where fast and reliable quality control is essential.

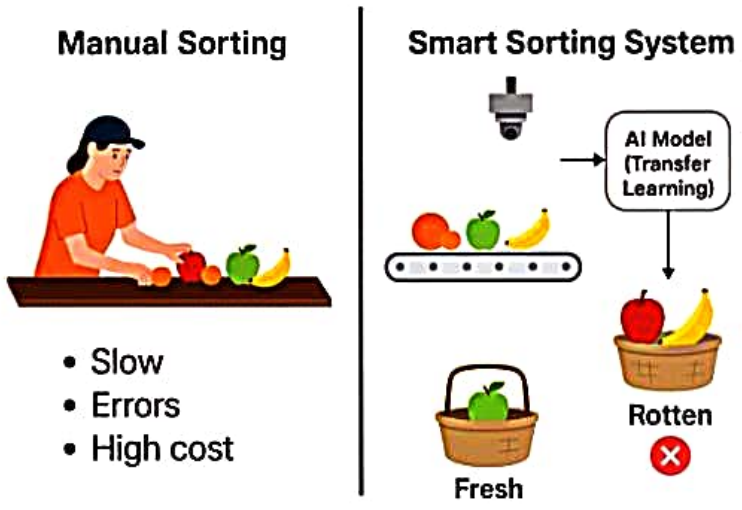
Problem Statement

Manual fruit and vegetable sorting has several challenges:
- Time-consuming and inefficient
- High risk of human error and oversight
- Physically exhausting for workers
- High labor costs
- Poor hygiene due to manual handling

To solve these, our system uses computer vision and transfer learning to automate the detection of rotten fruits. By using pre-trained deep learning models, we ensure accurate classification with less data and training time.

Real-World Example

In a fruit packaging warehouse, a camera captures images of fruits on a conveyor belt. The trained AI model analyzes the image and classifies it as fresh or rotten. Based on this result, a mechanical arm sorts the fruits accordingly. This ensures speed, consistency, and hygiene.



Objectives

1. To Automate the Quality Assessment Process

Develop a system that can automatically inspect and evaluate the quality of fruits and vegetables without human intervention. This helps reduce manual effort, saves time, and enhances consistency in sorting.

2. To Detect and Classify Fresh vs Rotten Produce Using Images

Use deep learning techniques to analyze visual features (like color, texture, shape) from images and classify each item as either fresh or rotten.

3. To Implement Transfer Learning for Accurate Classification

Leverage pre-trained models (e.g., MobileNet, ResNet) using transfer learning to improve classification performance even with a smaller dataset, speeding up training and improving accuracy.

4. To Enhance Efficiency and Reduce Sorting Errors

Replace manual sorting, which is slow and error-prone, with a faster and more reliable AI-based system that minimizes human error.

5. To Reduce Wastage and Post-Harvest Losses

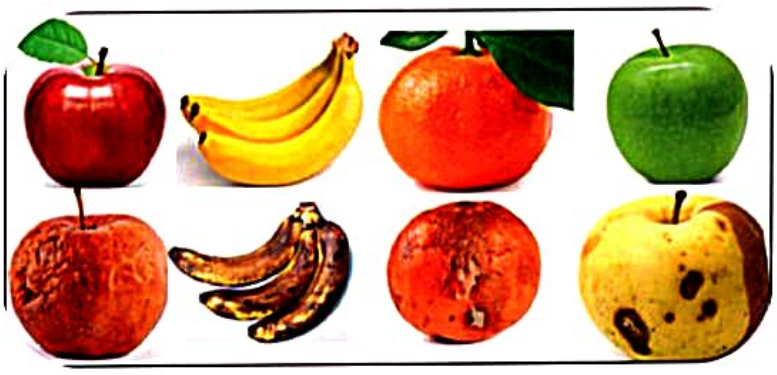
Quickly identifying and separating rotten items prevents spoilage of surrounding produce and ensures only fresh fruits reach consumers, reducing food wastage and loss.

6. To Enable Scalable Sorting in Large Agricultural and Retail Environments

Design a solution that can be deployed in real-world settings like farms, warehouses, and supermarkets, and is scalable for high-volume sorting.

7. To Build a Cost-Effective Alternative to Manual Labor

Reduce operational costs associated with hiring and training human labor by implementing a reusable AI-based solution.



Proposed solution/methodology

Dataset Used

A fruit and vegetable image dataset is used that contains labeled images of fresh and rotten produce. These labels help train the model to distinguish between healthy and spoiled items.

Pre-processing Steps

Before feeding images into the model, several pre-processing steps are applied:
- Resizing images to a standard size
- Normalizing pixel values to improve training stability
- Data augmentation (rotation, flipping, brightness adjustment) to improve generalization.

Model Used

Pre-trained CNN architectures like MobileNet, ResNet, or VGG are used. These models are well-suited for image classification tasks and serve as strong base models for transfer learning.

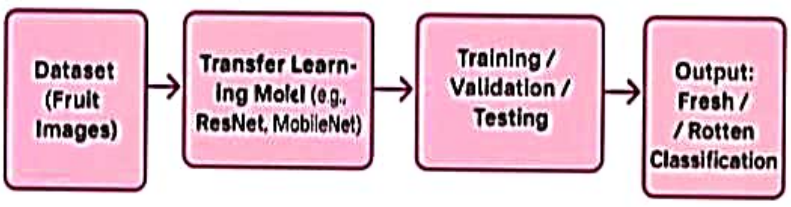
Transfer Learning Concept

Transfer learning involves taking a pre-trained model (e.g., trained on ImageNet) and reusing its weights to train on our specific fruit dataset. This helps improve accuracy even with a smaller dataset and reduces training time significantly.

Training, Validation, and Testing Phases

The dataset is divided into three parts:
- Training set: Used to teach the model
- Validation set: Used to tune hyperparameters and prevent overfitting
- Testing set: Used to evaluate the model's final performance on unseen data.

METHODOLOGY FLOWCHART



Project flow explanation

1. Dataset

The dataset contains images of various fruits, such as apples and bananas. Each image is labeled as either fresh or rotten. These labeled images are used as input to train the model to identify the condition of fruits automatically.

2. Data Preprocessing

Before feeding the data into the model, preprocessing is done to enhance performance:
- Resizing: All images are resized to a uniform size (e.g., 224x224 pixels).
- Normalization: Pixel values are scaled to the range 0–1.
- Data Augmentation: Techniques such as flipping, rotating, or zooming help increase diversity and improve generalization.

3. Pre-trained CNN Model

A pre-trained convolutional neural network (CNN) like MobileNet or ResNet is used. These models are trained on large datasets and are used as feature extractors, providing high-level image features to the classifier without starting from scratch.

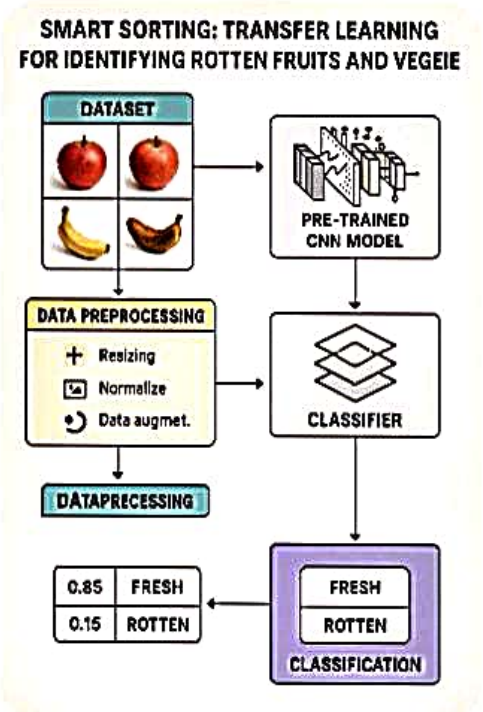
4. Classifier

The output from the CNN is passed through a custom classifier. The structure typically includes a Flatten layer, Dense layers with ReLU activation, and a final Softmax layer to classify the image as fresh or rotten.

5. Classification

The model produces a probability for each class. For example:
- Fresh: 0.85
- Rotten: 0.15

The label with the highest probability is considered the final classification output.



Smart Sorting: Transfer Learning Architecture

1. Input Layer

- **What it does:** Takes in an image of a fruit or vegetable.
- **Image size:** Typically resized to 224x224 pixels with 3 color channels (RGB).
- **Why:** Standard size required by pre-trained models like ResNet or MobileNet.

2. Preprocessing Layer

- **What it does:** Normalizes pixel values and resizes images.
- **Why:** To match the input format expected by the pre-trained model and improve performance.

3. Pre-trained CNN Base Model (e.g., MobileNet / ResNet)

- **What it does:** Extracts deep features from the image.
- **How it works:**
- - Convolutional Layers detect edges, shapes, and patterns.
- - Pooling Layers reduce the spatial dimensions.
- **Why:** Leverages a model trained on a large dataset (ImageNet) to save time and resources.

4. Flatten / Global Average Pooling Layer

- **What it does:** Converts the high-dimensional features into a 1D vector.
- **Why:** Prepares the features for the fully connected layers.

5. Dense (Fully Connected) Layer

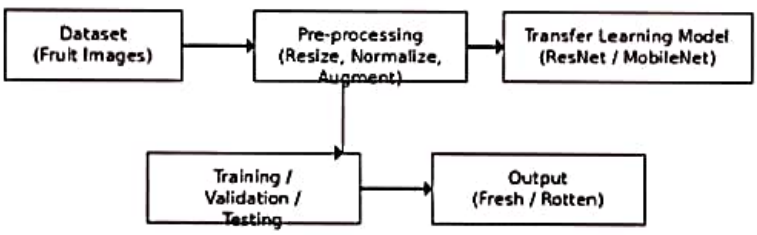
- **What it does:** Learns to associate extracted features with specific classes.
- **Neurons:** Typically 64 or 128 (can vary).
- **Activation:** ReLU for non-linearity.

6. Output Layer

- **What it does:** Predicts whether the fruit/vegetable is Fresh or Rotten.
- **Activation Function:** Softmax (for probability distribution).
- **Classes:** Fresh (0) or Rotten (1)

Summary

- This architecture applies transfer learning by reusing a CNN trained on general images, then fine-tuning it for fruit and vegetable classification.
- This saves training time, improves accuracy, and works well even with smaller datasets.



Results and Output

Smart Sorting uses AI and machine learning techniques to automatically identify and classify fruits and vegetables as either fresh or rotten. By training a deep learning model with images of different produce, the system learns patterns and features (such as color, texture, and shape) that indicate spoilage. This solution is faster and more reliable than manual sorting, especially when scaled to large quantities in real-time applications like food processing plants and supermarkets.

Summary of Results:

The trained model achieved promising results in detecting rotten fruits and vegetables. Evaluation metrics like accuracy, loss, and confusion matrix were used to assess its performance. Visual outputs also show successful classification of images, which indicates the practical usefulness of the model.

Accuracy and Loss Graphs:

- Accuracy represents how often the model correctly classifies the images.
- Loss indicates the model's error or deviation from the correct answer.
- During training, we expect the accuracy to increase and loss to decrease over time.
- These graphs help to visualize the training process and detect underfitting or overfitting.

Confusion Matrix:

- A confusion matrix is a table that summarizes the performance of a classification model.
- It shows True Positives, True Negatives, False Positives, and False Negatives.
- This matrix helps in identifying specific areas where the model may be making errors.

Sample Outputs:

- These are images from the test set, processed by the model.
- Each image is labeled by the model as either Fresh or Rotten.
- These visual results help verify if the model is working as expected.

9. Code Snapshot

Model Definition:

```
model = tf.keras.Sequential([
  base_model,
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation='relu'),
```

Output will be in the form of


```
[40]: predict = model.predict(img_bat)
      1/1 [==============================] - 0s 29ms/step
[41]: score = tf.nn.softmax(predict)
[42]: print('Veg/Fruit in image is {} with accuracy of {:0.2f}'.format(data_cat[np.argmax(score)],np.max(score)*100))
      Veg/Fruit in image is RottenTomato with accuracy of 100.00
```

```
3]: predict = model.predict(img_bat)
    score = tf.nn.softmax(predict)
    print('Veg/Fruit in image is {} with accuracy of {:0.2f}'.format(data_cat[np.argmax(score)],np.max(score)*100))

    1/1 [==============================] - 0s 27ms/step
    Veg/Fruit in image is FreshCucumber with accuracy of 100.00
```

```
  tf.keras.layers.Dense(2, activation='softmax')
])
```

Loading Dataset:

```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
  'data/train', image_size=(224, 224), batch_size=32
)
```

Prediction:

```
predictions = model.predict(test_images)
```

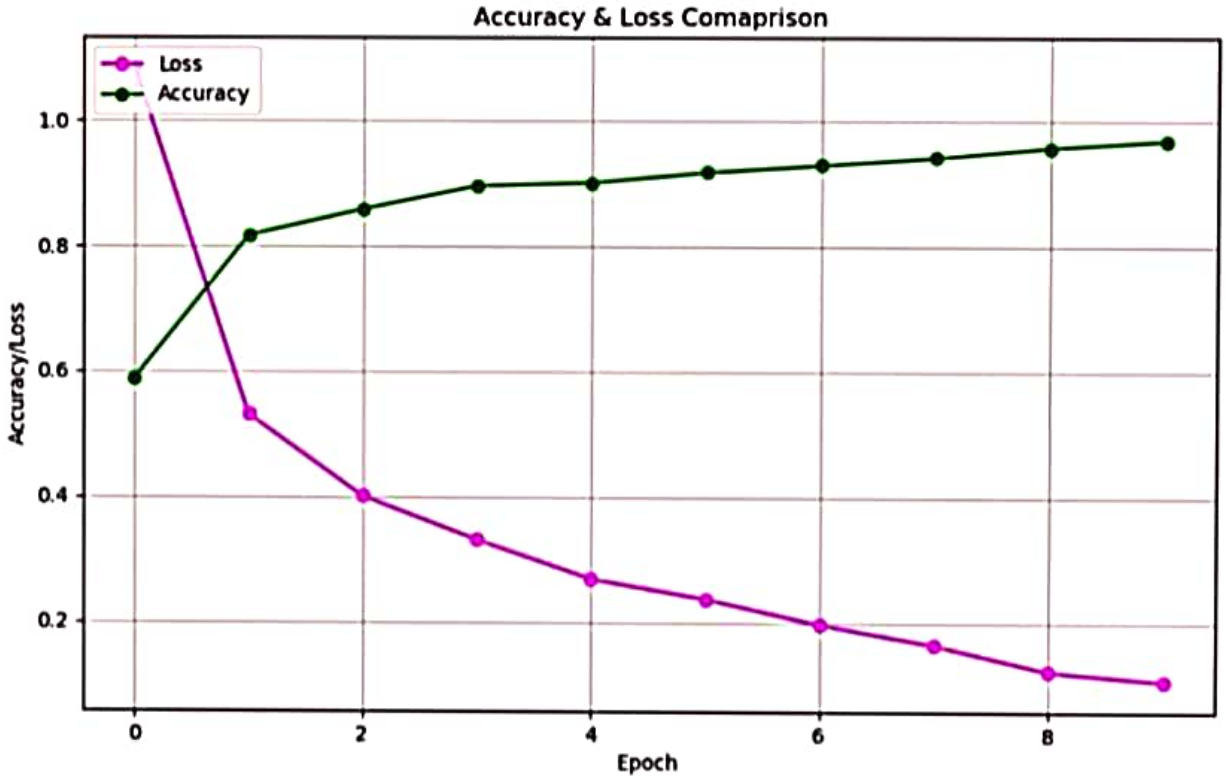


Figure 4: Accuracy and loss Comparison Graph

Objectives

- Develop an automated system to identify and classify rotten fruits and vegetables.
- Utilize transfer learning techniques for high accuracy and performance.
- Reduce human effort and improve efficiency in food processing and retail industries.

Advantages

- Reduces human error and increases accuracy.
- Speeds up the sorting process.
- Consistent and scalable solution.
- Cost-effective in the long run.
- Can be used in mobile or embedded systems.

Disadvantages

- Requires a large and diverse dataset.
- Accuracy can drop with poor-quality images.
- Needs regular retraining for new produce.
- Initial setup cost and technical knowledge required.

Challenges

- Collecting and labeling high-quality dataset.
- Ensuring real-time performance.
- Handling variety in shapes, sizes, and lighting.
- Integration into existing sorting machinery.
- Adapting to different environments (e.g., farms, markets).

Applications

- Supermarkets for automated quality check.
- Agricultural sorting centers.
- Food packaging industries.
- Waste reduction in supply chain.
- Integration into mobile apps for farmers.

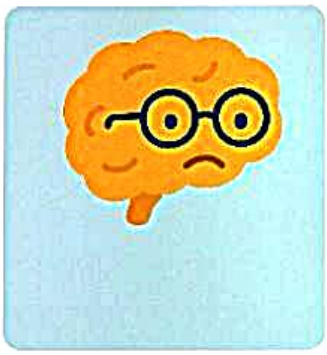
WHAT CAN IT DO?

✓ IDENTIFY ROTTEN APPLES

✓ CAPTION IMAGES

✓ TRANSLATI TEXT

WHAT CAN'T IT DO?



✗ WRITE A BOOK

✗ DO MATH HOMEWORK

✗ TELL THE FUTURE