**Algorithm Overview:**

The model uses **Transfer Learning** with a **MobileNetV2** backbone combined with custom convolutional and fully connected layers to solve a binary classification problem: predicting whether an image of a fruit/vegetable is "fresh" (class 0) or "rotten" (class 1).

**Working Steps:**

1. **Transfer Learning with MobileNetV2**:
   * **MobileNetV2** is a lightweight convolutional neural network designed for mobile and embedded vision applications.
   * Here, we load pre-trained weights of MobileNetV2 to leverage its feature extraction capabilities.
   * The model is configured to exclude the fully connected "top" layers (include\_top=False) since we will add custom layers for our specific task.
2. **Freezing Pre-trained Layers**:
   * The pre-trained layers of MobileNetV2 are frozen to prevent their weights from being updated during training.
   * This helps retain the general image feature extraction capability learned from the original training on a large dataset (like ImageNet).
3. **Custom Layers for Specific Classification**:
   * Custom convolutional layers are added to further refine features relevant to the "fresh vs. rotten" classification.
   * Layers include:
     + **SeparableConv2D**: Efficient convolutional layers that separate spatial and depth-wise convolutions, reducing computational load.
     + **BatchNormalization**: Normalizes activations to stabilize training and improve generalization.
     + **Dropout**: Adds randomness to prevent overfitting.
     + **Dense**: Fully connected layers for classification.
4. **Activation Functions**:
   * **ReLU** (Rectified Linear Unit) is used in intermediate layers to introduce non-linearity.
   * **Sigmoid** is used in the output layer to provide a probability score between 0 (fresh) and 1 (rotten).
5. **Compilation**:
   * The model is compiled with:
     + **Binary Cross-Entropy**: A loss function used for binary classification problems.
     + **Adam Optimizer**: An adaptive learning rate optimizer that adjusts learning rates during training.
6. **Learning Rate Scheduler**:
   * **ReduceLROnPlateau** reduces the learning rate if the validation loss plateaus, allowing the model to fine-tune itself during training.
7. **Model Checkpoint**:
   * Saves the best version of the model based on validation loss, ensuring the final model is the most effective.

**Data Pipeline:**

1. **Input Data**:
   * Images of fruits and vegetables, resized to a fixed size of 100×100×3100 \times 100 \times 3100×100×3.
   * Each image is associated with a label (0 for fresh, 1 for rotten).
2. **Training and Validation**:
   * The dataset is split into training and validation sets.
   * The model learns from the training set and evaluates its performance on the validation set.
3. **Predictions**:
   * A single input image is reshaped to 1×100×100×31 \times 100 \times 100 \times 31×100×100×3 and passed through the model.
   * The model outputs a probability score:
     + Close to **0** indicates "fresh."
     + Close to **1** indicates "rotten."

**Problem-Specific Challenges Addressed:**

1. **Class Imbalance**:
   * If the dataset is skewed (e.g., more fresh images than rotten), techniques like class weighting or data augmentation could help.
2. **Feature Extraction**:
   * MobileNetV2 extracts high-level features (e.g., texture, color differences) critical for distinguishing fresh and rotten produce.
3. **Overfitting**:
   * Regularization techniques like Dropout and BatchNormalization reduce overfitting, ensuring the model generalizes well to unseen data.

**Visualization:**

The final training and validation performance are visualized through loss and accuracy plots, giving insights into the model's learning progress.