Introduction

The goal of this project is to build a model to classify fruits and vegetables using images. The two different approaches are CNN, to see how well it can learn to classify the images, and transfer learning with MobileNetV2, a pre trained model, to see if we can get results faster. By comparing these two methods, I hope to find the best way to build an accurate and efficient model.

Analysis

The data set used contains 36 different classes of fruits and vegetables. It can be found here

Fruits: Banana, Apple, Kiwi, Pomegranate, Grapes, Mango, Orange, Pineapple, Pear, Watermelon

Vegetables: Lettuce, Onion, Peas, Ginger, Turnip, Beetroot, Cucumber, Jalapeno, Potato, Chilli Pepper, Corn, Sweetcorn, Cauliflower, Carrot, Eggplant, Spinach, Capiscum, Radish, Bell Pepper, Sweet Potato, Lemon, Cabbage, Paprika, Tomato, Soybean, Garlic

Data was split into 3 major categories: Train, Test, and Validation. It was split 3115 : 359 : 351 respectively.

The images were all resized to 128x128 and pixels were normalized in order to help stabilize the training process and potentially lead to faster convergence and better model performance.

Methods

Convolutional Neural Network (CNN)

1. **Input Preprocessing**:
    - Images were resized to (128, 128) and rescaled by dividing pixel values by 255.0 to normalize them within the range [0, 1].
2. **Architecture**:
    - **Rescaling Layer**: Normalizes the input images to improve training stability.
    - **Convolutional Block 1**:
        - Conv2D layer with 32 filters, a (3x3) kernel, ReLU activation, and L2 regularization to prevent overfitting.
        - MaxPooling2D layer with a (2x2) pool size to reduce spatial dimensions.
    - **Convolutional Block 2**:

        - Conv2D layer with 64 filters, a (3x3) kernel, ReLU activation, and L2 regularization.
        - MaxPooling2D layer with a (2x2) pool size.
    - **Convolutional Block 3**:
        - Conv2D layer with 128 filters, a (3x3) kernel, ReLU activation, and L2 regularization.
        - MaxPooling2D layer with a (2x2) pool size.
    - **Convolutional Block 4**:
        - Conv2D layer with 128 filters, a (3x3) kernel, ReLU activation, and L2 regularization.
        - MaxPooling2D layer with a (2x2) pool size.
    - **Convolutional Block 5**:
        - Conv2D layer with 256 filters, a (3x3) kernel, ReLU activation, and L2 regularization.
        - Flatten layer to convert the 2D feature maps into a 1D vector.
    - **Fully Connected Layers**:
        - Dense layer with 512 units, ReLU activation, and L2 regularization.
        - Dropout layer with a rate of 0.5 to mitigate overfitting.
        - Dense output layer with a softmax activation to output class probabilities.
3. **Compilation**:
    - Optimizer: Adam with a learning rate of 0.0001 for efficient gradient descent.
    - Loss Function: categorical_crossentropy suitable for multi-class classification.
    - Metrics: accuracy to track model performance during training.
4. **Training**:
    - The model was trained for 50 epochs using the training and validation datasets.
    - Early stopping was not explicitly used but can be added for regularization.

Transfer Learning with MobileNetV2

1. **Pre-trained Model**:
    - Used MobileNetV2, a pre-trained model on ImageNet, as the feature extractor.
    - The model was loaded without the top layers (include_top=False), and weights were frozen to retain pre-trained features.
2. **Custom Model Head**:
    - **Input Layer**: Accepts images resized to (128, 128, 3) and uses MobileNetV2's preprocessing function.
    - **Feature Extraction**: MobileNetV2 processes the input images but does not update its weights during training.
    - **GlobalAveragePooling2D**: Reduces the feature maps into a single vector per image, minimizing overfitting and computation.
    - **Dense Layer**: 512 units with ReLU activation and L2 regularization.
    - **Dropout Layer**: 0.5 dropout rate to prevent overfitting.
    - **Output Layer**: Softmax activation function for classifying images into the target categories.

3. **Compilation**:
    - Optimizer: Adam with a learning rate of 0.0001.
    - Loss Function: categorical_crossentropy for multi-class classification.
    - Metrics: accuracy to measure performance.
4. **Training**:
    - The model was trained for 5 epochs due to MobileNetV2's efficiency in extracting features.
    - This shorter training time significantly reduced computational costs while still yielding good performance.

Results

CNN Model Performance

- **Runtime**: 21 minutes and 28 seconds
- **Final Training Loss**: 0.8599
- **Training Accuracy**: 87.96%
- **Final Validation Loss**: 0.7596
- **Validation Accuracy**: 92.88%
- **Test Loss**: 0.7547
- **Test Accuracy**: 93.04%

Metrics Discussion:

- The CNN model took much longer to train compared to the Transfer Learning model. However, it achieved very good performance metrics, with a high test accuracy of 93.04%.
- The relatively close values of training loss and validation loss suggest that the model generalizes well to the validation data, indicating minimal overfitting.
- The high accuracy and low loss values indicate that the CNN effectively learned to distinguish between classes in the dataset, but the runtime is a concern for efficiency.

Notable Issues or Patterns:

- The CNN model is computationally expensive.
- Could be optimized further to reduce runtime.

Transfer Learning Model (MobileNetV2) Performance

- **Runtime**: 1 minute and 26 seconds
- **Final Training Loss**: 1.6693
- **Training Accuracy**: 72.71%
- **Final Validation Loss**: 1.1355
- **Validation Accuracy**: 89.74%
- **Test Loss**: 1.1416
- **Test Accuracy**: 89.69%

Metrics Discussion:

- The Transfer Learning model using MobileNetV2 had a much shorter runtime
- The training accuracy of 72.71% is lower compared to the CNN, indicating that the model did not learn as well during training. However, the validation and test accuracy values suggest that MobileNetV2 generalized well to unseen data.

Notable Issues or Patterns:

- The initial training accuracy is lower

Comparison Between Models

1. The CNN model achieved a higher test accuracy compared to the Transfer Learning model .
2. The Transfer Learning model trained almost 15 times faster than the CNN model
3. Both models generalized well to the test data

Reflection

1. I would experiment with reducing the number of filters and layers to strike a better balance between accuracy and runtime
2. In future projects, especially those with large datasets or limited time, I would prioritize using pre-trained models for rapid development. Transfer learning offers a great starting point

Final Thoughts:

This assignment demonstrated the power of transfer learning for providing a practical approach that saves time and computational power