Vegetable Classification using Transfer Learning: A Comparative Study of VGG16 and ResNet50 Models

Keerthi Devireddy
Dept. of Computer Science
University of North Carolina at Greensboro
Cary, USA
Email: k_devireddy@uncg.edu

Abstract—**In this study, I investigate the application of transfer learning techniques using the VGG16 and ResNet50 pre-trained models for the task of vegetable classification. The primary objective is to leverage the powerful feature extraction capabilities of the deep neural networks, pretrained on the ImageNet dataset, and fine-tune them for accurate classification of vegetable images. The work explores the performance of both models on a dataset consisting of 10 vegetable classes, evaluating their accuracy, loss, and qualitative predictions. Through comprehensive experiments and analysis, I aim to provide insights into the strengths and limitations of each model, facilitating informed decision-making for future vegetable classification tasks. The study highlights the efficacy of transfer learning.**

I. Introduction

A. Background and Motivation

The ability to accurately classify and identify different types of vegetables has significant implications in various domains, including agriculture, food processing, and dietary analysis. With the emergence of deep learning and computer vision techniques, automating the process of vegetable classification has become increasingly viable. However, developing robust and accurate models from scratch can be challenging, particularly when dealing with limited labeled data or computational resources.

B. Problem Statement

The primary objective of this research is to evaluate the performance of the VGG16 and ResNet50 models, when fine-tuned on a vegetable image dataset, for the task of classifying different types of vegetables. By conducting a comparative analysis, I aim to identify the strengths and weaknesses of each model, providing insights into their suitability for vegetable classification tasks.

C. Objectives

The specific objectives of the work are as follows:

1) To fine-tune and adapt the VGG16 and ResNet50 models for vegetable classification using transfer learning techniques.
2) To assess the performance of the fine-tuned models in terms of classification accuracy, loss, and qualitative predictions.
3) To conduct a comparative analysis between the VGG16 and ResNet50 models, highlighting their respective strengths and limitations.

II. Related Work

A. Brief review of relevant work

In this mini project, I focused on exploring and leveraging two well-established convolutional neural network (CNN) architectures, VGG16 and ResNet50, for the task of vegetable classification. These models have been extensively studied and employed in various computer vision and image recognition tasks, demonstrating remarkable performance and serving as strong baselines for transfer learning applications.
The VGG16 model, introduced by Simonyan and Zisserman, 2014, is a deep CNN architecture that achieved outstanding results in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Its key characteristics include the use of small 3x3 convolutional filters and a deep stack of convolutional layers, enabling the extraction of rich hierarchical features from input images .
On the other hand, the ResNet50 model, proposed by Heetal, 2016, introduced the concept of residual connections, which alleviated the vanishing gradient problem in deep neural networks. This innovative architecture allowed for the training of significantly deeper models, leading to improved performance on various image recognition tasks, including the ILSVRC challenge.
Both VGG16 and ResNet50 have been widely adopted in transfer learning scenarios, where the models are pre-trained on large-scale datasets, such as ImageNet, and then fine-tuned on specific target tasks. This approach leverages the learned feature representations from the pre-training phase, allowing for faster convergence and better performance, even with limited training data.

B. Existing Approaches and Methods for Vegetable Classification

In my exploration of vegetable classification, I began by gathering a suitable dataset containing images of different vegetable classes. This involved removing unwanted images and cleaning the data, ensuring proper labeling and organization of the images.
To gain a comprehensive understanding of the VGG16 and ResNet50 models, I studied their architectural details, including the number of layers, types of convolutional and pooling operations, and the use of specific techniques like residual connections and small convolutional filters. Additionally, I explored the transfer learning approach, which involves fine-tuning pre-trained models on a specific task, in this case, vegetable classification.
Prior to implementing the models, I familiarized myself with the concepts of data augmentation, which is crucial for improving model generalization and preventing overfitting. Techniques such as rotation, flipping, and scaling were considered to increase the diversity of the training data.
Furthermore, I researched various optimization algorithms, loss functions, and regularization techniques commonly used in CNN training, as these components play a vital role in achieving optimal model performance.

III. Theoretical Background

VGG16 Architecture

The VGG16 architecture, is a prominent CNN model widely used for image classification tasks. It consists of 16 convolutional layers arranged in a specific pattern, followed by fully connected layers. The VGG16 model was pre-trained on the ImageNet dataset, a large-scale dataset containing over 1 million images and 1,000 object categories with 92.7% accuracy. This pre-training allows the model to learn robust and generalizable features, which can be transferred and fine-tuned for various computer vision tasks. This model differed from previous high-performing models in several ways. First, it used a tiny 3×3 receptive field with a 1-pixel stride—for comparison, AlexNet used an 11×11 receptive field with a 4-pixel stride. The 3×3 filters combine to provide the function of a larger receptive field. The benefit of using multiple smaller layers rather than a single large layer is that more non-linear activation layers accompany the convolution layers, improving the decision functions and allowing the network to converge quickly. Second, VGG uses a smaller convolutional filter, which reduces the network's tendency to over-fit during training exercises. A 3×3 filter is the optimal size because a smaller size cannot capture left-right and up-down information. Thus, VGG is the smallest possible model to understand an image's spatial features. Consistent 3×3 convolutions make the network easy to manage.

1) Input: VGGNet receives a 224×224 image input. In the ImageNet competition, the model's creators kept the image input size constant by cropping a 224×224 section from the center of each image.

2) Convolutional layers: the convolutional filters of VGG use the smallest possible receptive field of 3×3. VGG also uses a 1×1 convolution filter as the input's linear transformation.

3) ReLu activation: next is the Rectified Linear Unit Activation Function (ReLU) component, AlexNet's major innovation for reducing training time. ReLU is a linear function that provides a matching output for positive inputs and outputs zero for negative inputs. VGG has a set convolution stride of 1 pixel to preserve the spatial resolution after convolution (the stride value reflects how many pixels the filter "moves" to cover the entire space of the image).

4) Hidden layers : all the VGG network's hidden layers use ReLU instead of Local Response Normalization like AlexNet. The latter increases training time and memory consumption with little improvement to overall accuracy.

5) Pooling layers : A pooling layer follows several convolutional layers—this helps reduce the dimensionality and the number of parameters of the feature maps created by each convolution step. Pooling is crucial given the rapid growth of the number of available filters from 64 to 128, 256, and eventually 512 in the final layers.

6) Fully connected layers : VGGNet includes three fully connected layers. The first two layers each have 4096 channels, and the third layer has 1000 channels, one for every class.

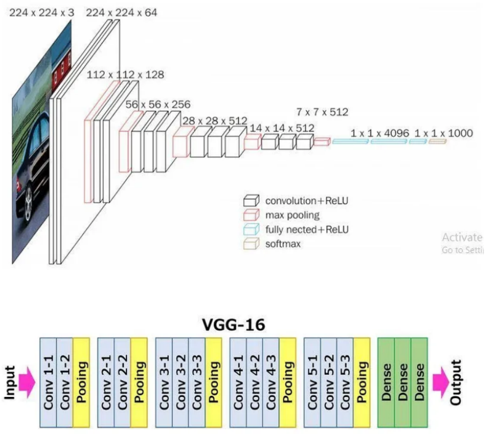


Fig. 1. VGG16 Architecture

RESNET50 Architecture

ResNet-50 consists of 50 layers that are divided into 5 blocks, each containing a set of residual blocks. The residual blocks allow for the preservation of information from earlier layers, which helps the network to learn better representations of the input data.
The following are the main components of ResNET.

1) Convolutional Layers: The first layer of the network is a convolutional layer that performs convolution on the input image. This is followed by a max-pooling layer that downsamples the output of the convolutional layer. The output of the max-pooling layer is then passed through a series of residual blocks.

2) Residual Blocks: Each residual block consists of two convolutional layers, each followed by a batch normalization layer and a rectified linear unit (ReLU) activation function. The output of the second convolutional layer is then added to the input of the residual block, which is then passed through another ReLU activation function. The output of the residual block is then passed on to the next block.
3) Fully Connected Layer: The final layer of the network is a fully connected layer that takes the output of the last residual block and maps it to the output classes. The number of neurons in the fully connected layer is equal to the number of output classes.

Key Features of ResNet-50

- ILSVRC'15 classification winner (3.57% top 5 error)
- 152 layer model for ImageNet
- Has other variants also (with 35, 50, 101 layers)
- Every 'residual block' has two 3×3 convolution layers
- No FC layer, except one last 1000 FC softmax layer for classification
- Global average pooling layer after the last convolution
- Batch Normalization after every convolution layer
- SGD + momentum (0.9)
- No dropout used

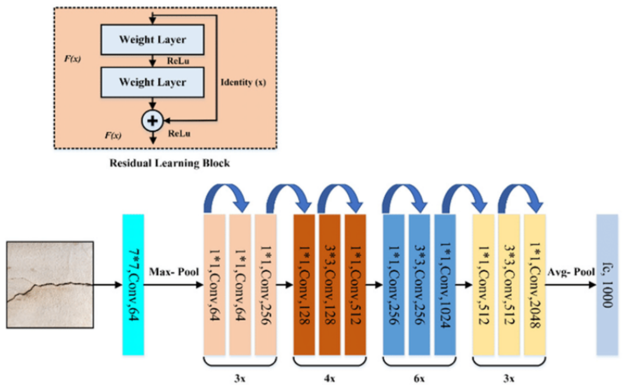


Fig. 2. ResNet-50 Architecture

Advantages of ResNet-50 Over Other Networks:
ResNet-50 has several advantages over other networks. One of the main advantages is its ability to train very deep networks with hundreds of layers.
This is made possible by the use of residual blocks and skip connections, which allow for the preservation of information from earlier layers.
Another advantage of ResNet-50 is its ability to achieve state-of-the-art results in a wide range of image-related tasks such as object detection, image classification, and image segmentation.
Conclusion:
In conclusion, ResNet-50 is a powerful and efficient deep neural network that has revolutionized the field of computer vision. Its architecture, skip connections, and advantages over other networks make it an ideal choice for a wide range of image-related tasks.

IV. Dataset

A. Description of the Data Used

For this mini project, I utilized a dataset consisting of vegetable images from 10 different classes: Broccoli, Cabbage, Capsicum, Carrot, Cauliflower, Cucumber, Potato, Pumpkin, Radish, Tomato. The dataset was obtained from Kaggle, which is a publicly available resource for computer vision and image recognition tasks.
The dataset comprised a total of 6500 images, with 5000 images allocated for training, 750 images for validation, and 750 images for testing purposes. The images in the dataset varied in terms of resolution, orientation, and lighting conditions, reflecting the diversity and real-world scenarios encountered in vegetable classification tasks.

B. Data Preprocessing Steps

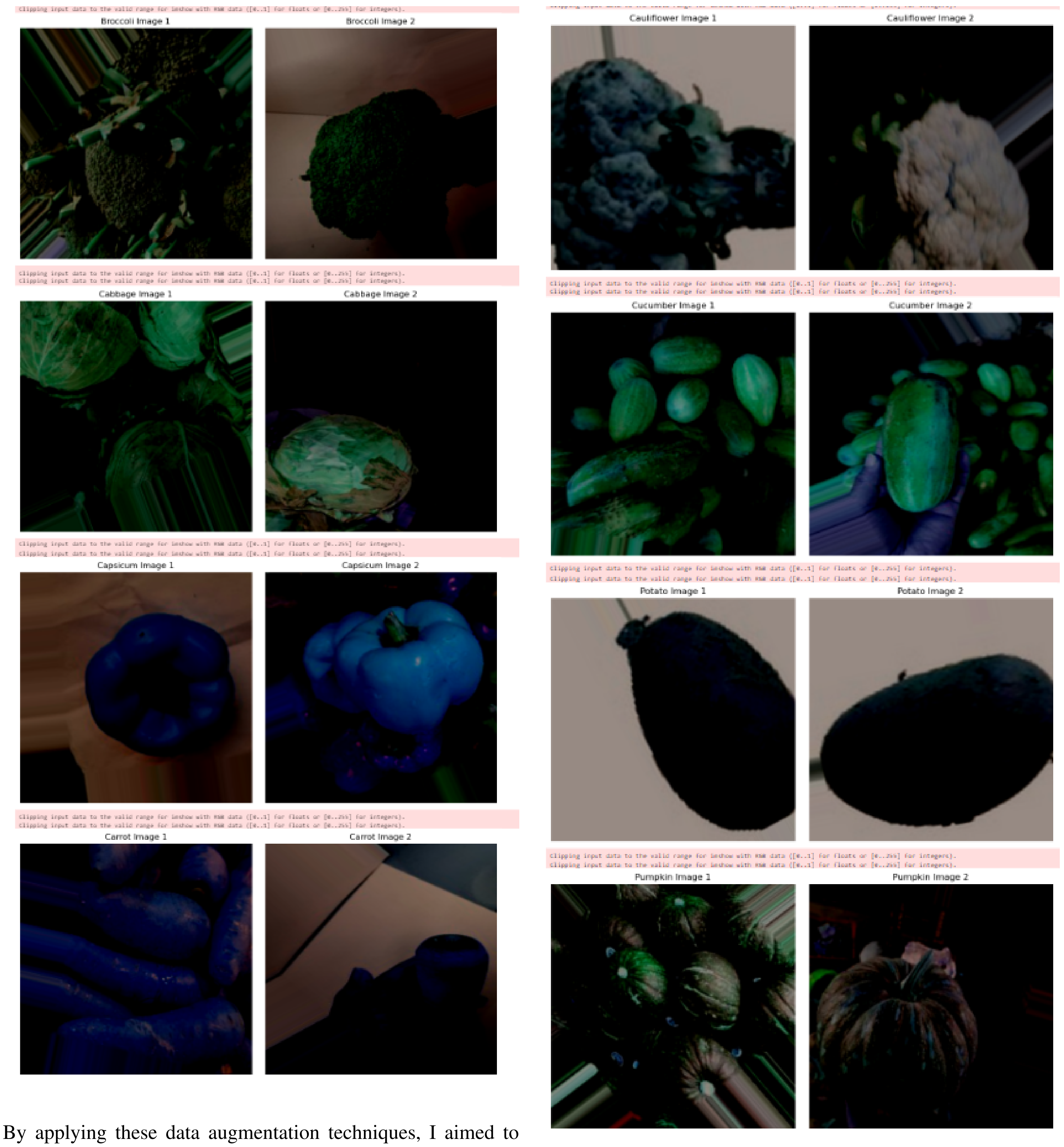
Before proceeding with model training, I performed several data preprocessing steps to ensure data quality and consistency. These steps included:

Image resizing: To maintain a consistent input size for the CNN models, I resized all images to target resolution, e.g., 224x224 pixels. Data splitting: I divided the dataset into training, validation, and test sets following the aforementioned distribution (70:15:15). This split ensured that the models were evaluated on unseen data during the testing phase, providing an unbiased assessment of their performance. Data normalization: To improve the training stability and convergence, I normalized the pixel values of the images to a range of 0 to 1 Label encoding: Since the CNN models require numerical labels for classification, I encoded the vegetable class names into corresponding numerical labels using one-hot encoding.

C. Data Augmentation Techniques Applied

To increase the diversity of the training data and improve model generalization, I employed various data augmentation techniques. These techniques artificially generate new training samples by applying transformations to the existing images, thereby exposing the models to a broader range of variations. The specific data augmentation techniques used in this project were:

1) Rotation: I applied random rotations to the images within a range of 40. This helped the models learn rotational invariance and handle variations in vegetable orientation.
2) Horizontal and vertical flipping: I randomly flipped the images horizontally and vertically, which is particularly useful for vegetable classes that exhibit symmetry or lack a preferred orientation.
3) Scaling and zooming: I performed random scaling and zooming operations on the images within a range of 0.2. This simulated variations in the distance from which the vegetables were captured.
4) Brightness and contrast adjustments: I applied random adjustments to the brightness and contrast levels of the images, mimicking different lighting conditions and helping the models learn robust feature representations.



```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

Fig. 3. Data Augmentation

By applying these data augmentation techniques, I aimed to create a more diverse and robust training set, enabling the models to better generalize to unseen data and handle real-world variations in vegetable images.

V. Methodology

A. *Transfer Learning Approach*

In this project, I leveraged the power of transfer learning to adapt the pre-trained VGG16 and ResNet50 models, originally trained on the ImageNet dataset, for the task of vegetable classification. Transfer learning involves fine-tuning a pre-trained model on a specific target task, taking advantage of the learned feature representations from the initial training phase. The transfer learning approach I followed consisted of the following steps:

1) Freezing the Base Model: I froze the weights of the pre-

trained base models (VGG16 and ResNet50), ensuring that the learned feature representations from the ImageNet dataset were preserved.

2) Adding Custom Layers: I added custom fully connected layers on top of the frozen base models, tailored for the vegetable classification task. These layers were initialized with random weights and would be trained during the fine-tuning process.
3) Fine-tuning: I fine-tuned the custom layers added to the base models using the vegetable dataset. During this process, the weights of the base models remained frozen, while the weights of the custom layers were updated to learn the specific features relevant to vegetable classification.

By leveraging transfer learning, I aimed to take advantage of the powerful feature representations learned by the pre-trained models on a large-scale dataset, while efficiently adapting them to the specific task of vegetable classification, even with a relatively small dataset.

B. Model Architecture and Modifications

For both the VGG16 and ResNet50 models, I made several modifications to adapt them to the vegetable classification task:

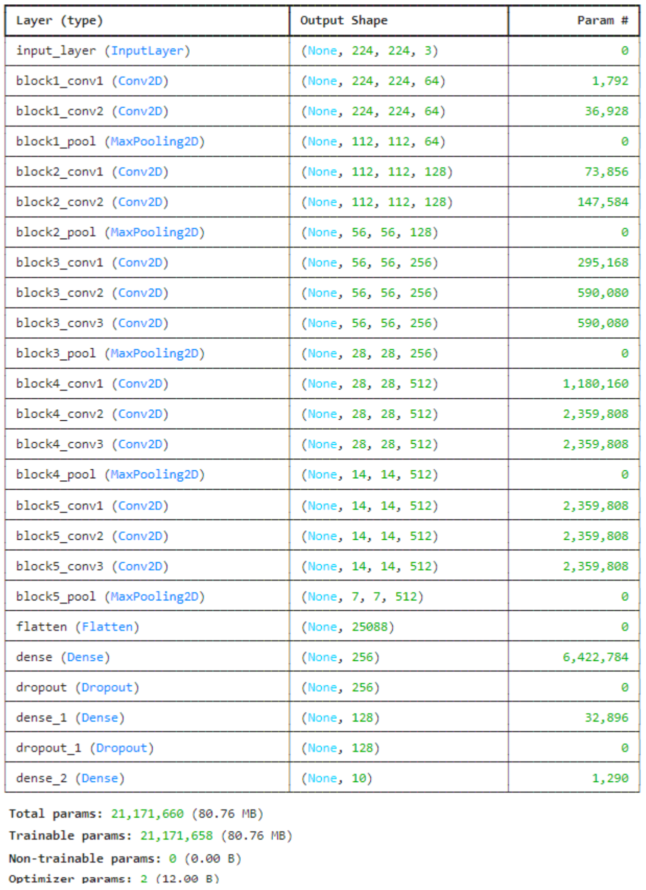
1) VGG16 Model:

- I removed the final fully connected layers from the pre-trained VGG16 model, as they were originally designed for the ImageNet classification task.
- I added a global average pooling layer to reduce the spatial dimensions of the feature maps to a single vector.
- I introduced three new fully connected layers: the first layer with 256 units, followed by a dropout layer with a rate of 0.5 to mitigate overfitting, the second layer with 128 units and another dropout layer with a rate of 0.5, and finally, the output layer with 10 units, corresponding to the number of vegetable classes.
- I applied L2 regularization with a factor of 0.001 to the weights of the first and second fully connected layers to further prevent overfitting.

2) ResNet50 Model:

- Similar to the VGG16 model, I removed the final fully connected layer from the pre-trained ResNet50 model.
- I added a global average pooling layer to reduce the spatial dimensions of the feature maps to a single vector.
- I introduced three new fully connected layers: the first layer with 256 units, followed by a dropout layer with a rate of 0.5, the second layer with 128 units and another dropout layer with a rate of 0.5, and finally, the output layer with 10 units.
- I applied L2 regularization with a factor of 0.001 to the weights of the first and second fully connected layers.

By making these modifications, I aimed to adapt the pre-trained models to the specific requirements of the vegetable classification task, while leveraging their powerful feature extraction capabilities.



```
Layer (type)                     Output Shape                      Param #
input_layer (InputLayer)         (None, 224, 224, 3)                     0
block1_conv1 (Conv2D)            (None, 224, 224, 64)                1,792
block1_conv2 (Conv2D)            (None, 224, 224, 64)               36,928
block1_pool (MaxPooling2D)       (None, 112, 112, 64)                    0
block2_conv1 (Conv2D)            (None, 112, 112, 128)              73,856
block2_conv2 (Conv2D)            (None, 112, 112, 128)             147,584
block2_pool (MaxPooling2D)       (None, 56, 56, 128)                     0
block3_conv1 (Conv2D)            (None, 56, 56, 256)               295,168
block3_conv2 (Conv2D)            (None, 56, 56, 256)               590,080
block3_conv3 (Conv2D)            (None, 56, 56, 256)               590,080
block3_pool (MaxPooling2D)       (None, 28, 28, 256)                     0
block4_conv1 (Conv2D)            (None, 28, 28, 512)             1,180,160
block4_conv2 (Conv2D)            (None, 28, 28, 512)             2,359,808
block4_conv3 (Conv2D)            (None, 28, 28, 512)             2,359,808
block4_pool (MaxPooling2D)       (None, 14, 14, 512)                     0
block5_conv1 (Conv2D)            (None, 14, 14, 512)             2,359,808
block5_conv2 (Conv2D)            (None, 14, 14, 512)             2,359,808
block5_conv3 (Conv2D)            (None, 14, 14, 512)             2,359,808
block5_pool (MaxPooling2D)       (None, 7, 7, 512)                       0
flatten (Flatten)                (None, 25088)                           0
dense (Dense)                    (None, 256)                     6,422,784
dropout (Dropout)                (None, 256)                             0
dense_1 (Dense)                  (None, 128)                        32,896
dropout_1 (Dropout)              (None, 128)                             0
dense_2 (Dense)                  (None, 10)                          1,290

Total params: 21,171,660 (80.76 MB)
Trainable params: 21,171,658 (80.76 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 2 (12.00 B)
```

Fig. 4. Customised VGG16 Model Summary

C. Training Process

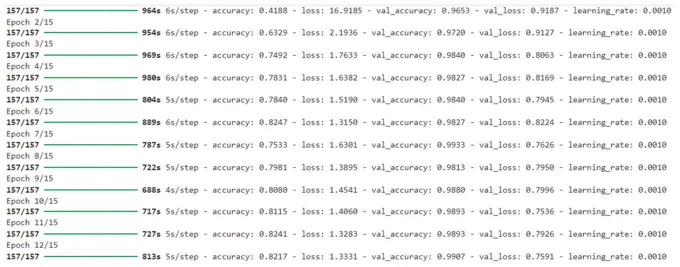
During the training process, I employed several techniques and configurations to optimize the performance of the models:

1) Optimizer: I used the Adam optimizer with a learning rate of 0.001 to update the weights of the custom layers during fine-tuning. The Adam optimizer is a popular choice for its adaptive learning rate and efficient convergence.
2) Loss Function: For the multi-class classification task, I utilized the categorical cross-entropy loss function, which is commonly used for measuring the performance of models with mutually exclusive classes.
3) To monitor and control the training process, I implemented several callbacks:
   - Early Stopping: This callback monitored the validation accuracy and stopped the training if no improvement was observed for a specified number of epochs (5 epochs), preventing overfitting and saving computational resources.
   - Reduce Learning Rate on Plateau: This callback reduced the learning rate by a factor of 0.2 if the validation loss did not improve for 3 consecutive epochs, enabling the model to navigate through potential plateaus and potentially find a better opti-

mum.

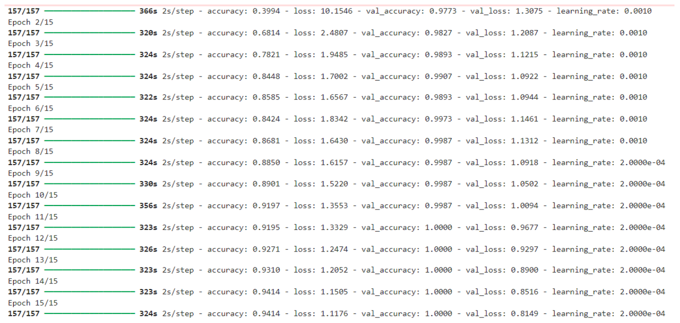
- Progress Callback: I implemented a custom callback to display the training progress percentage for each batch, providing visual feedback during the training process.

4) Data Augmentation: As mentioned in the dataset section, I applied various data augmentation techniques, including rotation, flipping, scaling, and brightness/contrast adjustments, to increase the diversity of the training data and improve model generalization.



```
157/157 ━━━━━━━━━━━━━━━━━━━━ 964s 6s/step - accuracy: 0.4188 - loss: 16.9185 - val_accuracy: 0.9653 - val_loss: 0.9187 - learning_rate: 0.0010
Epoch 2/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 954s 6s/step - accuracy: 0.6329 - loss: 2.1936 - val_accuracy: 0.9720 - val_loss: 0.9127 - learning_rate: 0.0010
Epoch 3/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 969s 6s/step - accuracy: 0.7492 - loss: 1.7633 - val_accuracy: 0.9840 - val_loss: 0.8063 - learning_rate: 0.0010
Epoch 4/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 980s 6s/step - accuracy: 0.7831 - loss: 1.6382 - val_accuracy: 0.9827 - val_loss: 0.8169 - learning_rate: 0.0010
Epoch 5/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 804s 5s/step - accuracy: 0.7840 - loss: 1.5190 - val_accuracy: 0.9840 - val_loss: 0.7945 - learning_rate: 0.0010
Epoch 6/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 889s 6s/step - accuracy: 0.8247 - loss: 1.3150 - val_accuracy: 0.9827 - val_loss: 0.8224 - learning_rate: 0.0010
Epoch 7/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 787s 5s/step - accuracy: 0.7533 - loss: 1.6301 - val_accuracy: 0.9933 - val_loss: 0.7626 - learning_rate: 0.0010
Epoch 8/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 722s 5s/step - accuracy: 0.7981 - loss: 1.3895 - val_accuracy: 0.9813 - val_loss: 0.7950 - learning_rate: 0.0010
Epoch 9/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 688s 4s/step - accuracy: 0.8080 - loss: 1.4541 - val_accuracy: 0.9880 - val_loss: 0.7996 - learning_rate: 0.0010
Epoch 10/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 717s 5s/step - accuracy: 0.8115 - loss: 1.4060 - val_accuracy: 0.9893 - val_loss: 0.7536 - learning_rate: 0.0010
Epoch 11/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 727s 5s/step - accuracy: 0.8241 - loss: 1.3283 - val_accuracy: 0.9893 - val_loss: 0.7926 - learning_rate: 0.0010
Epoch 12/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 813s 5s/step - accuracy: 0.8217 - loss: 1.3331 - val_accuracy: 0.9907 - val_loss: 0.7591 - learning_rate: 0.0010
```

Fig. 5. VGG16 Training Progress



```
157/157 ━━━━━━━━━━━━━━━━━━━━ 366s 2s/step - accuracy: 0.3994 - loss: 10.1546 - val_accuracy: 0.9773 - val_loss: 1.3075 - learning_rate: 0.0010
Epoch 2/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 320s 2s/step - accuracy: 0.6814 - loss: 2.4807 - val_accuracy: 0.9827 - val_loss: 1.2087 - learning_rate: 0.0010
Epoch 3/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 324s 2s/step - accuracy: 0.7821 - loss: 1.9485 - val_accuracy: 0.9893 - val_loss: 1.1215 - learning_rate: 0.0010
Epoch 4/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 324s 2s/step - accuracy: 0.8448 - loss: 1.7002 - val_accuracy: 0.9907 - val_loss: 1.0922 - learning_rate: 0.0010
Epoch 5/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 322s 2s/step - accuracy: 0.8585 - loss: 1.6567 - val_accuracy: 0.9893 - val_loss: 1.0944 - learning_rate: 0.0010
Epoch 6/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 324s 2s/step - accuracy: 0.8424 - loss: 1.8342 - val_accuracy: 0.9973 - val_loss: 1.1461 - learning_rate: 0.0010
Epoch 7/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 324s 2s/step - accuracy: 0.8681 - loss: 1.6430 - val_accuracy: 0.9987 - val_loss: 1.1312 - learning_rate: 0.0010
Epoch 8/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 324s 2s/step - accuracy: 0.8850 - loss: 1.6157 - val_accuracy: 0.9987 - val_loss: 1.0918 - learning_rate: 2.0000e-04
Epoch 9/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 330s 2s/step - accuracy: 0.8901 - loss: 1.5220 - val_accuracy: 0.9987 - val_loss: 1.0502 - learning_rate: 2.0000e-04
Epoch 10/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 356s 2s/step - accuracy: 0.9197 - loss: 1.3553 - val_accuracy: 0.9987 - val_loss: 1.0094 - learning_rate: 2.0000e-04
Epoch 11/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 323s 2s/step - accuracy: 0.9195 - loss: 1.3329 - val_accuracy: 1.0000 - val_loss: 0.9677 - learning_rate: 2.0000e-04
Epoch 12/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 326s 2s/step - accuracy: 0.9271 - loss: 1.2474 - val_accuracy: 1.0000 - val_loss: 0.9297 - learning_rate: 2.0000e-04
Epoch 13/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 323s 2s/step - accuracy: 0.9310 - loss: 1.2052 - val_accuracy: 1.0000 - val_loss: 0.8900 - learning_rate: 2.0000e-04
Epoch 14/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 323s 2s/step - accuracy: 0.9414 - loss: 1.1505 - val_accuracy: 1.0000 - val_loss: 0.8516 - learning_rate: 2.0000e-04
Epoch 15/15
157/157 ━━━━━━━━━━━━━━━━━━━━ 324s 2s/step - accuracy: 0.9414 - loss: 1.1176 - val_accuracy: 1.0000 - val_loss: 0.8149 - learning_rate: 2.0000e-04
```

Fig. 6. ResNet50 Training Progress

By carefully selecting and configuring these components, I aimed to optimize the training process, prevent overfitting, and achieve the best possible performance for the vegetable classification task.

VI. Experimental Setup

A. Hardware and Software Specifications

Executed the mini project on my personal computer with the following specifications:

- Hardware: 11th Gen Intel Core i5 processor, 8GB RAM
- Operating System: Windows 11

B. Implementation Details

The code for this project was implemented using Jupyter Notebook, leveraging the following Python libraries TensorFlow, Keras, NumPy, Matplotlib, OpenCV.

VII. Results and Discussion

A. Training and Validation Performance (Accuracy and Loss Curves)

During the training process, I monitored the accuracy and loss metrics for both the VGG16 and ResNet50 models on the training and validation datasets. The following figures illustrate the progression of these metrics over the training epochs:

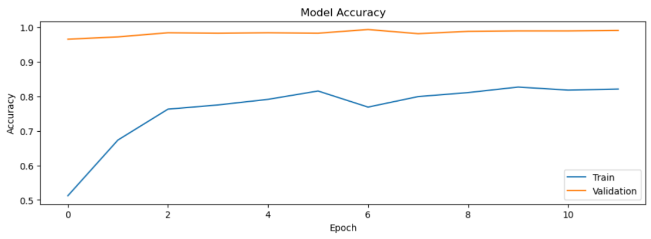


Fig. 7. VGG16 Training and Validation Accuracy Curves

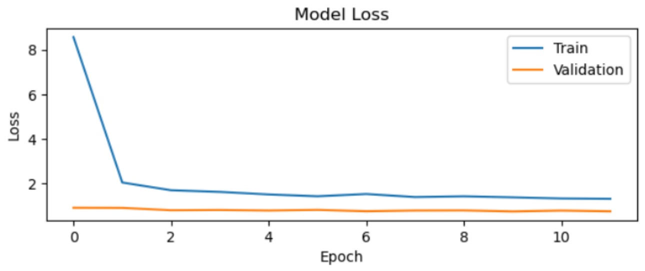


Fig. 8. VGG16 Training and Validation Loss Curves

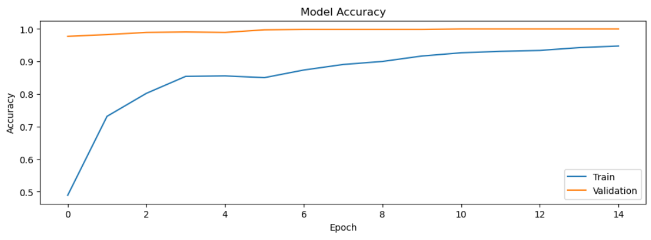


Fig. 9. ResNet50 Training and Validation Accuracy Curves

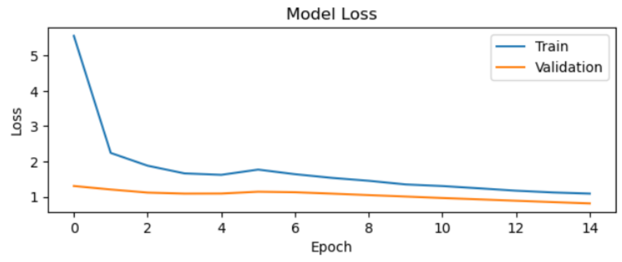


Fig. 10. ResNet50 Training and Validation Loss Curves

1. VGG16:
Starting from zero the training accuracy seems to be linearly increasing till epoch 5 and then there is a slight drop at epoch 6 and has raised a bit and has constantly increased till epoch 15. Lasting around 82% after the final run.
While the loss had consistently dropped till epoch 1 and from there on it remained stable throughout all the epochs execution (flat line).
Coming to the Validation accuracy and loss curves, the validation accuracy curves pretty much seemed to be a straight line with just a minor up and downs at few places. After the final run the validation accuracy lasted arounf 99%.
In the same way, the validation loss curve seemed to be a flat line at the with a very minor ups and downs at few places. 2. RestNet50:
Coming to ResNet50, the training accuracy curve is similar to that of the vgg16 ones, but the percentage obtained at the end is higher than that of the vgg16, the curve seemed to be linearly increasing till epoch 5 and then there is a slight drop at epoch 6 and has raised a bit and has a constant linear increase till epoch 15. Lasting around 94% after the final run.
While the loss had consistently dropped till epoch 1 and from there on it remained stable throughout all the epochs execution (flat line).
Coming to the Validation accuracy and loss curves, the validation accuracy curves pretty much seemed to be a straight line with just a minor up and downs at few places. After the final run the validation accuracy lasted around 99%.
In the same way, the validation loss curve linearly decreased till epoch 10 and remained constant from there on.

B. Test Set Performance

After training the models, I evaluated their performance on the unseen test dataset. The following table summarizes the test set accuracy and loss for both the VGG16 and ResNet50 models:

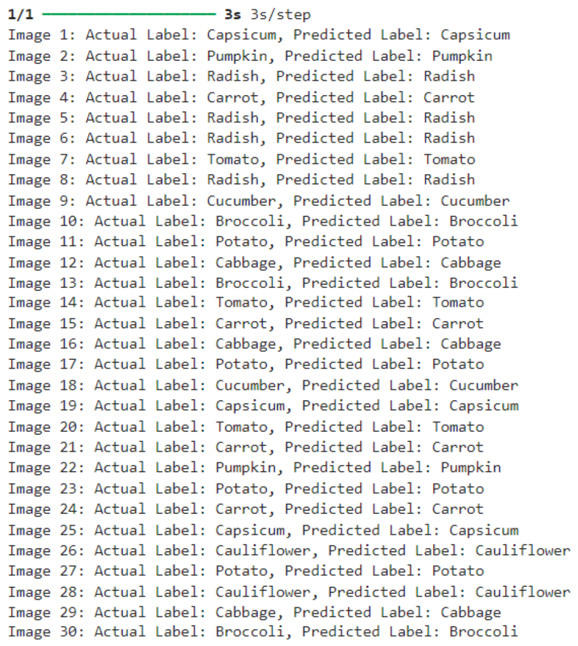
Model	Accuracy	Loss
VGG16	97.47%	0.79%
ResNet50	99.73%	0.98%

C. Qualitative Analysis

In addition to quantitative metrics, it is essential to examine the qualitative performance of the models by analyzing sample predictions. The following examples illustrate the predictions made by the VGG16 and ResNet50 models on selected test images:

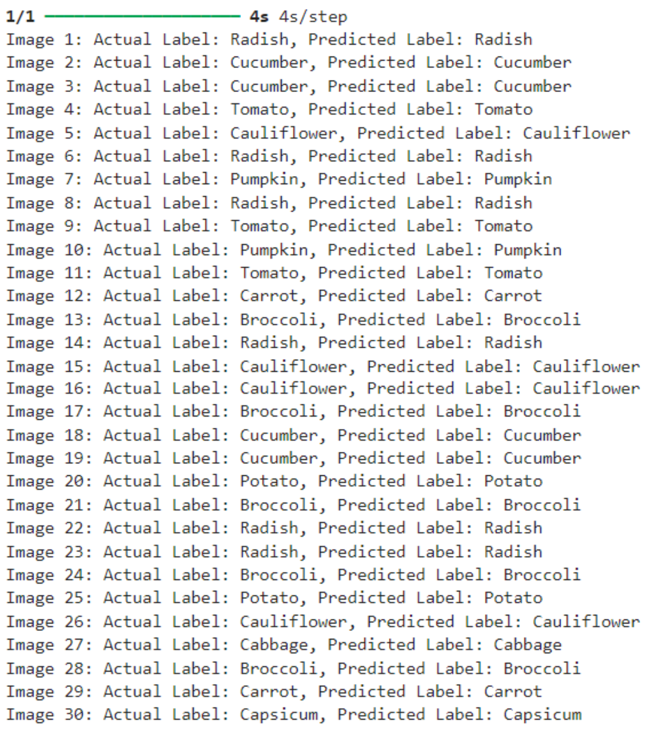
D. Comparison of VGG16 and ResNet50 Models

- Accuracy: The ResNet50 model achieved a higher test set accuracy of 99.73% compared to the VGG16 model's accuracy of 97.47%. This suggests that the ResNet50 architecture may be more suitable for the vegetable classification task, potentially due to its deeper and more complex architecture, which can capture more intricate features.



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step
Image 1: Actual Label: Capsicum, Predicted Label: Capsicum
Image 2: Actual Label: Pumpkin, Predicted Label: Pumpkin
Image 3: Actual Label: Radish, Predicted Label: Radish
Image 4: Actual Label: Carrot, Predicted Label: Carrot
Image 5: Actual Label: Radish, Predicted Label: Radish
Image 6: Actual Label: Radish, Predicted Label: Radish
Image 7: Actual Label: Tomato, Predicted Label: Tomato
Image 8: Actual Label: Radish, Predicted Label: Radish
Image 9: Actual Label: Cucumber, Predicted Label: Cucumber
Image 10: Actual Label: Broccoli, Predicted Label: Broccoli
Image 11: Actual Label: Potato, Predicted Label: Potato
Image 12: Actual Label: Cabbage, Predicted Label: Cabbage
Image 13: Actual Label: Broccoli, Predicted Label: Broccoli
Image 14: Actual Label: Tomato, Predicted Label: Tomato
Image 15: Actual Label: Carrot, Predicted Label: Carrot
Image 16: Actual Label: Cabbage, Predicted Label: Cabbage
Image 17: Actual Label: Potato, Predicted Label: Potato
Image 18: Actual Label: Cucumber, Predicted Label: Cucumber
Image 19: Actual Label: Capsicum, Predicted Label: Capsicum
Image 20: Actual Label: Tomato, Predicted Label: Tomato
Image 21: Actual Label: Carrot, Predicted Label: Carrot
Image 22: Actual Label: Pumpkin, Predicted Label: Pumpkin
Image 23: Actual Label: Potato, Predicted Label: Potato
Image 24: Actual Label: Carrot, Predicted Label: Carrot
Image 25: Actual Label: Capsicum, Predicted Label: Capsicum
Image 26: Actual Label: Cauliflower, Predicted Label: Cauliflower
Image 27: Actual Label: Potato, Predicted Label: Potato
Image 28: Actual Label: Cauliflower, Predicted Label: Cauliflower
Image 29: Actual Label: Cabbage, Predicted Label: Cabbage
Image 30: Actual Label: Broccoli, Predicted Label: Broccoli
```

Fig. 11. VGG16 Test Predictions



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step
Image 1: Actual Label: Radish, Predicted Label: Radish
Image 2: Actual Label: Cucumber, Predicted Label: Cucumber
Image 3: Actual Label: Cucumber, Predicted Label: Cucumber
Image 4: Actual Label: Tomato, Predicted Label: Tomato
Image 5: Actual Label: Cauliflower, Predicted Label: Cauliflower
Image 6: Actual Label: Radish, Predicted Label: Radish
Image 7: Actual Label: Pumpkin, Predicted Label: Pumpkin
Image 8: Actual Label: Radish, Predicted Label: Radish
Image 9: Actual Label: Tomato, Predicted Label: Tomato
Image 10: Actual Label: Pumpkin, Predicted Label: Pumpkin
Image 11: Actual Label: Tomato, Predicted Label: Tomato
Image 12: Actual Label: Carrot, Predicted Label: Carrot
Image 13: Actual Label: Broccoli, Predicted Label: Broccoli
Image 14: Actual Label: Radish, Predicted Label: Radish
Image 15: Actual Label: Cauliflower, Predicted Label: Cauliflower
Image 16: Actual Label: Cauliflower, Predicted Label: Cauliflower
Image 17: Actual Label: Broccoli, Predicted Label: Broccoli
Image 18: Actual Label: Cucumber, Predicted Label: Cucumber
Image 19: Actual Label: Cucumber, Predicted Label: Cucumber
Image 20: Actual Label: Potato, Predicted Label: Potato
Image 21: Actual Label: Broccoli, Predicted Label: Broccoli
Image 22: Actual Label: Radish, Predicted Label: Radish
Image 23: Actual Label: Radish, Predicted Label: Radish
Image 24: Actual Label: Broccoli, Predicted Label: Broccoli
Image 25: Actual Label: Potato, Predicted Label: Potato
Image 26: Actual Label: Cauliflower, Predicted Label: Cauliflower
Image 27: Actual Label: Cabbage, Predicted Label: Cabbage
Image 28: Actual Label: Broccoli, Predicted Label: Broccoli
Image 29: Actual Label: Carrot, Predicted Label: Carrot
Image 30: Actual Label: Capsicum, Predicted Label: Capsicum
```

Fig. 12. ResNet50 Test predictions

- Convergence Speed: From the training logs, it can be observed that the ResNet50 model converged faster than the VGG16 model, reaching higher accuracy and lower loss values in fewer epochs. This could be attributed to the residual connections in ResNet50, which help mitigate the vanishing gradient problem and enable more efficient training of deeper networks.

- Computational Efficiency: The ResNet50 model is generally more computationally efficient than the VGG16 model, owing to its use of bottleneck blocks and residual connections, which reduce the number of parameters and computational requirements. However, a detailed analysis of inference times and resource utilization would be required for a more conclusive comparison.

E. Discussions of Results and Potential Imporvements

Strengths and Limitations: Both the VGG16 and ResNet50 models demonstrated impressive performance in the vegetable classification task, achieving high accuracy on the test set. However, the ResNet50 model outperformed the VGG16 model in terms of overall accuracy and convergence speed. The depth and residual connections of the ResNet50 architecture enabled it to capture more complex features and mitigate the vanishing gradient problem during training. Potential Improvements: To further enhance the performance of these models, several improvements could be explored:

- Fine-tuning deeper layers: While I froze the base model layers during transfer learning, fine-tuning some of the deeper layers could potentially improve the feature representations specific to the vegetable classification task.
- Ensemble methods: Combining the predictions of multiple models, such as VGG16 and ResNet50, through ensemble techniques like bagging or boosting, could potentially improve the overall accuracy and robustness of the system.
- Data augmentation strategies: Experimenting with additional data augmentation techniques, such as elastic deformations, color jittering, or mixup, could help increase the diversity of the training data and improve model generalization.
- Hyperparameter tuning: Conducting a systematic hyperparameter search or employing techniques like Bayesian optimization could lead to optimal configurations of learning rates, regularization strengths, and other hyperparameters, potentially boosting model performance.

Future Directions: Some potential future directions for this work could include:

- Exploration of other CNN architectures: While VGG16 and ResNet50 are powerful models, exploring and evaluating the performance of other state-of-the-art CNN architectures, such as EfficientNets or Vision Transformers, could yield further insights and improvements.
- Incorporation of additional modalities: Combining visual information from images with other modalities, such as textual descriptions or sensor data, could enhance the vegetable classification capabilities and enable more robust and comprehensive systems.
- Transfer learning from larger datasets: Pre-training the models on larger and more diverse datasets, such as the ImageNet-21K or the Google Open Images dataset, could potentially improve the learned feature representations and facilitate better transfer to the vegetable classification task.

VIII. Conclusion

A. Summary of the Work

In this mini project, I explored the application of transfer learning techniques using the VGG16 and ResNet50 pretrained models for the task of vegetable classification. The primary objective was to leverage the powerful feature extraction capabilities of these deep neural networks, originally trained on the ImageNet dataset, and fine-tune them to accurately classify vegetable images from a dataset consisting of 10 different classes.

B. Key Findings

1) Both the VGG16 and ResNet50 models demonstrated impressive performance in the vegetable classification task, achieving high accuracy on the test set, with the ResNet50 model outperforming the VGG16 model in terms of overall accuracy (99.73% vs. 97.47%).
2) The ResNet50 model exhibited faster convergence during training, likely due to its residual connections, which help mitigate the vanishing gradient problem in deep neural networks.
3) The ResNet50 architecture is generally more computationally efficient than VGG16, owing to its use of bottleneck blocks and residual connections, which reduce the number of parameters and computational requirements. Potential improvements for future work include fine-tuning deeper layers, employing ensemble methods, exploring advanced data augmentation strategies, conducting hyperparameter tuning, and investigating other state-of-the-art CNN architectures or multi-modal approaches.

Overall, this mini project showcases the effectiveness of transfer learning and the remarkable performance of deep learning models, such as VGG16 and ResNet50, in the domain of vegetable classification.

References

[1] https://datagen.tech/guides/computer-vision/vgg16/
[2] https://wisdomml.in/understanding-resnet-50-in-depth-architecture-skip-connections-and-advantages-over-other-networks/