Transfer Learning: Fine-Tuning ResNet50 and MobileNetV3

1. Introduction

Transfer learning is an important tool in deep learning to solve the problem of small datasets or insufficient training data, nowadays, a wide variety of CNN models have been introduced such a ResNet50 and MobileNetV3.

Kaiming He et al. proposed a residual learning framework that has made the training of deep neural networks easier and faster. One of the most successful residual networks that has achieved great results on the ImageNet dataset is ResNet-50 which is summarized in Table 1. However, ResNet50 has a complex architecture which makes it hard to implement on mobile devices.

Layer	Output Size	Details
conv1	112×112	7×7, 64, stride 2
conv2_x	56×56	[1×1,64 → 3×3,64 → 1×1,256] × 3
conv3_x	28×28	[1×1,128 → 3×3,128 → 1×1,512] × 4
conv4_x	14×14	[1×1,256 → 3×3,256 → 1×1,1024] × 6
conv5_x	7×7	[1×1,512 → 3×3,512 → 1×1,2048] × 3
Final	1×1	Global avg pool → 1000-d fc → softmax

Table 1. ResNet Architecture

The MobileNet models are based on depth-wise separable convolution instead of a standard convolution. Standard convolution performs the channel and spatial wise computation in one step while the depth-wise separable convolution splits the process into two steps: Point-wise convolution and depth-wise convolution. The depthwise convolution applies a filter to each channel of the image input then the pointwise convolution applies a 1 × 1 convolution to combine the outputs of the former layer. This leads to a drastic reduction in the computational power required and the model complexity.

The three versions of the MobileNet models has been improved ever since they were developed in 2017. The main purpose of the MobileNets was to implement a light CNN model on mobile devices with a reduced model size (<10 MB) and a reduced number of parameters.

MobileNetV3 was introduced in 2019 and it was developed by dropping complex layers and using H-swish function instead of standard ReLU to further increase the network efficiency and accuracy. Figure 1 summarizes the architecture of MobileNetV3.

However, in some cases MobileNet models must be trained for a large number of epochs to achieve a good accuracy.

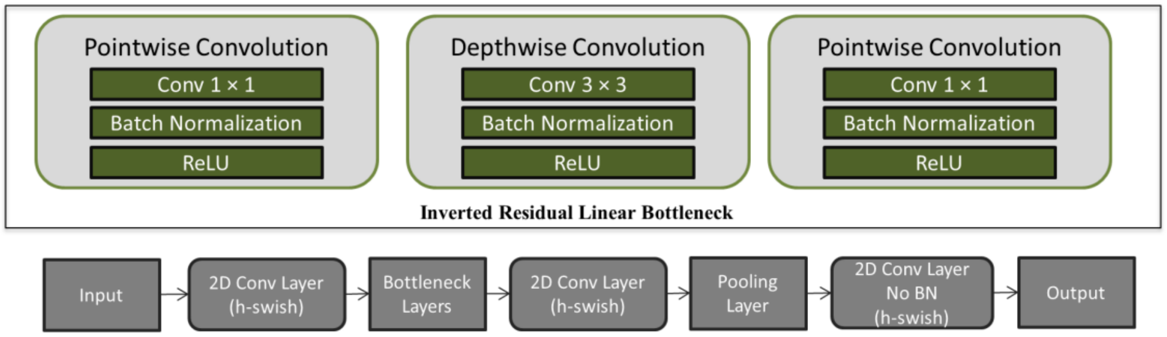

Figure 1 : MobileNetV3 Architecture

[1] **Optimized Deep Learning Algorithms for Tomato Leaf Disease Detection with Hardware Deployment Hesham Tarek 1,*, Hesham Aly 1 , Saleh Eisa 1 and Mohamed Abul-Soud 2**

2. Dataset

TF Flowers is a computer – vision dataset used for classifying 5 different types of flowers (dandelion , daisy, sunflowers, tulips, roses) .

Was split as:

- Training set with size of 2753 samples (75% of dataset).
- Validation set with size of 550 samples (15% of dataset).
- Testing set with size of 367   samples (10% of dataset).

With a total size of 3670 samples, which is considered as relatively small data size, especially for training deep models, therefore transfer learning will be applied to achieve strong performance despite the small size.

3. Methodology

3.1 Implementation

The methodology adopted used the TensorFlow GPU version as the primary library for building, training, and tuning models. Pre-trained models, such as ResNet50, MobileNetV3, and Xception with their weights set to ImageNet , were also downloaded from Keras, and data was imported from the TensorFlow Datasets library.

After importing the data and exploring some of the features and characteristics, it was discovered that the shape nature of the images was dynamic, not fixed. Some samples were displayed as an additional exploration step to further understand the nature of the data.

Next, Tensorflow's tf.data pipelines were constructed to batch and prefetch the data, and then preprocessing operations were performed using pre-trained applications (such as MobileNetV3 , ResNet50 preprocessing methods) to normalize and resize the images of each pretrained model. Following this approach, instead of converting the data to Numpy, yielded a significant performance difference, especially for the memory-intensive ResNet50 models due to it's complex architecture. Although these were pre-trained models, they consumed significant computing power and graphics card memory due to forward-passing operations despite freezing more layers during the tuning process, many memory exhausted errors occurred. Consequently, the pipeline approach allowed for more efficient training, as it divided the training, validation, and test batches into 32 batches, compared to the 16 used in the previous approach that utilized Numpy as input data. The model's performance and training time will be reviewed in the results section later.

Then, a compiling & training method was then built to train the fully frozen model, excluding the classification layers, as a primary baseline for measuring performance alongside models trained with advanced tuning, whether by block freezing or layer freezing.

Understanding the pretrained models and their complete structure is a fundamental step before beginning the advanced tuning & unfreezing process, determining the blocks or layers to freeze the model is essential. In our current dataset, given its small size, it is not preferable to unfreeze the deep layers of the model. The relationship between the size of the data you have and the number of layers or blocks to freeze is directly proportional to the number of layers or blocks to freeze. However, several experiments have been conducted to measure the actual performance and impact of unfreezing the deep layers.

3.2 Unfreezing Strategies

Two strategies were tested for fine-tuning:

- **Strategy A (Layer-wise)**: Unfreezing the last *n* layers of the base model to allow them to update during training.
- **Strategy B (Block-wise)**: Unfreezing entire blocks of layers based on the architectural boundaries of ResNet50 and MobileNetV3 , this required manual inspection of layer names and configurations.

In both strategies, a low learning rate (1e-5) was used for fine-tuning to prevent destabilizing the pre-trained weights, and each model was compiled using the Adam optimizer and categorical cross-entropy loss, and trained for 10 epochs.

Freezing models based on blocks is a somewhat easier approach and gives a quick impression of the results, but it does not offer the flexibility that tuning models based on layers provides. The initial layers closest to the input layer should remain frozen because they are well trained to capture edges and low-level features, which are essentially the same for all problems, therefore freezing the middle and upper layers was considered.

4. Results & Discussion

Here is a summary of the experiments that were tested on both models:

- Red Highlight indicates Fully Frozen Model.
- Green Highlight indicates Top Unfreezing by Layers Experiment.
- Yellow Highlight indicates Top Unfreezing by blocks Experiment.

Table 2. ResNet50 Pretrained Model Experiments

Experiment	Freezing Hyperparameters	Epochs	Model Type	Training Accuracy	Validation Accuracy	Test Accuracy
Exp 1	-	10	Full frozen model	67.68%	82.03%	83.65%
Exp 2	blocks_to_unfreeze = 2	5	Partially frozen (Blocks)	74.93%	72.96%	82.02%
Exp 3	unfreeze_from_layer = 56	10	Partially frozen (Layer Number)	94.50%	88.93%	92.64%
Exp 4	blocks_to_unfreeze = 10	10	Partially frozen (Blocks)	98.50%	80.22%	86.10%
Exp 5	unfreeze_from_layer = 100	10	Partially frozen (Layer Number)	94.06%	89.29%	89.37%

Exp 6	blocks_to_unfreeze = 3	10	Partially frozen (Blocks)	98.80%	90.20%	91.55%
Exp 7	unfreeze_from_layer = 120	10	Partially frozen (Layer Number)	95.20%	90.56%	91.83%
Exp 8	blocks_to_unfreeze = 8	10	Partially frozen (Blocks)	81.36%	71.87%	73.30%
Exp 9	unfreeze_from_layer = 80	10	Partially frozen (Layer Number)	95.11%	89.11%	91.28%
Exp 10	blocks_to_unfreeze = 15	10	Partially frozen (Blocks)	97.27%	84.21%	87.19%
Exp 11	unfreeze_from_layer = 150	10	Partially frozen (Layer Number)	94.10%	89.84%	91.28%
Exp 12	blocks_to_unfreeze = 20	10	Partially frozen (Blocks)	95.81%	80.58%	86.38%
Exp 13	unfreeze_from_layer = 7	10	Partially frozen (Layer Number)	94.24%	88.93%	90.19%
Exp 14	blocks_to_unfreeze = 1	10	Partially frozen (Blocks)	99.03%	88.57%	90.74%
Exp 15	unfreeze_from_layer = 170	10	Partially frozen (Layer Number)	94.70%	89.66%	91.55%

Table 3. MobileNetV3 Pretrained Model Experiments

Experiment	Freezing Hyperparameters	Epochs	Model Type	Training Accuracy	Validation Accuracy	Test Accuracy
Exp 1	-	10	Full frozen model	70.45%	85.84%	89.37%
Exp 2	unfreeze_from_layer = 56	5	Partially frozen (Blocks)	86.15%	84.57%	89.65%
Exp 3	blocks_to_unfreeze = 2	10	Partially frozen (Layer Number)	94.94%	89.47%	92.37%
Exp 4	blocks_to_unfreeze = 5	10	Partially frozen (Blocks)	98.71%	90.20%	93.19%
Exp 5	unfreeze_from_layer = 100	10	Partially frozen (Layer Number)	95.71%	89.47%	92.92%
Exp 6	blocks_to_unfreeze = 3	10	Partially frozen (Blocks)	99.48%	92.74%	94.28%
Exp 7	unfreeze_from_layer = 120	10	Partially frozen (Layer Number)	94.10%	89.66%	93.19%
Exp 8	blocks_to_unfreeze = 8	10	Partially frozen (Blocks)	98.68%	90.20%	93.46%
Exp 9	unfreeze_from_layer = 30	10	Partially frozen (Layer Number)	92.97%	88.57%	92.37%
Exp 10	blocks_to_unfreeze = 15	10	Partially frozen (Blocks)	97.26%	90.74%	92.64%

Experiment	**Freezing Hyperparameters**	**Epochs**	**Model Type**	**Training Accuracy**	**Validation Accuracy**	**Test Accuracy**
Exp 11	unfreeze_from_layer = 150	10	Partially frozen (Layer Number)	94.87%	90.38%	93.73%
Exp 12	blocks_to_unfreeze = 20	10	Partially frozen (Blocks)	98.17%	90.20%	93.73%
Exp 13	unfreeze_from_layer = 7	10	Partially frozen (Layer Number)	93.43%	90.93%	92.37%
Exp 14	blocks_to_unfreeze = 1	10	Partially frozen (Blocks)	98.21%	90.56%	93.46%
Exp 15	unfreeze_from_layer = 170	10	Partially frozen (Layer Number)	95.07%	89.84%	92.37%

As for the performance of the models in the case of completely freezing the layers (except for the classification layer), the models gave a modest result with a difference of about 6% in favor of MobileNetV3. The number of epochs , learning rate and the number of additional layers were fixed, so the focus was on adjusting the freezing hyperparameters (blocks and layers of each pretrained model) for a deeper comparison in the performance of the two models.

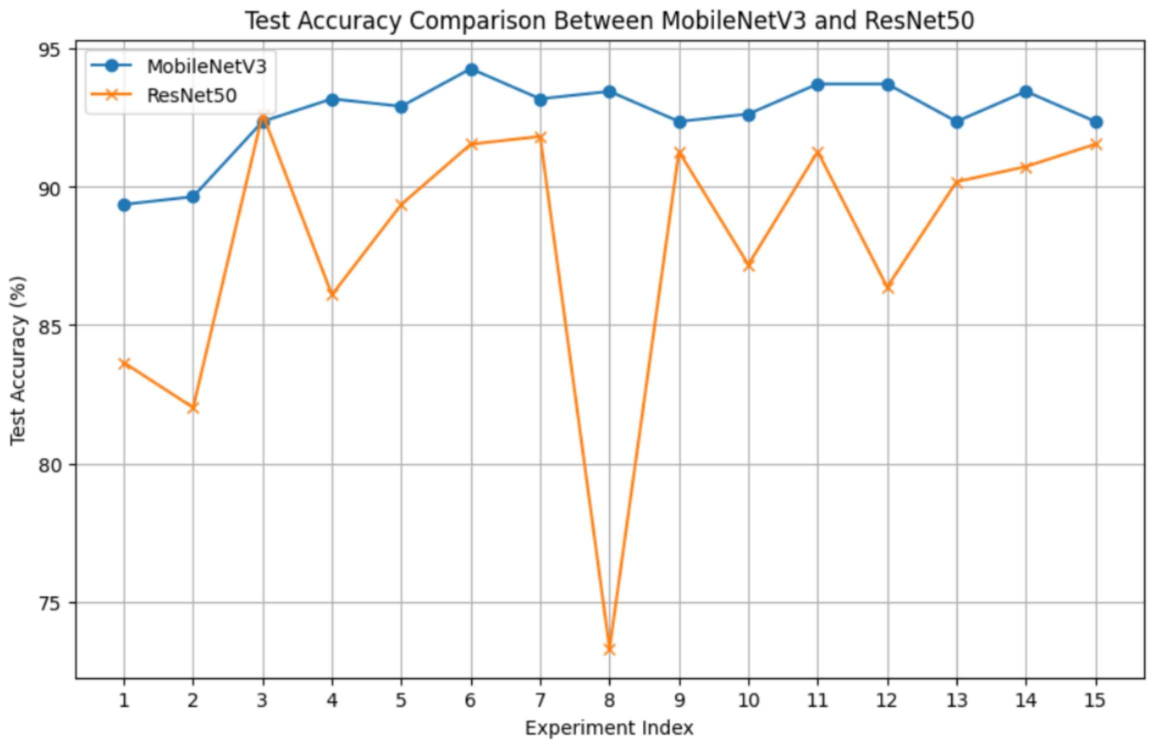

Figure 2 : Test Accuracy vs. Experiment Index for Different Unfreezing Strategies

The highest performance across all experiments was achieved using the **MobileNetV3** model with **3 blocks unfrozen** at experiment 6, reaching a **test accuracy of 94.28%** , for unfreezing layers approach , best result achieved in experiment 11 **with 93.73% test accuracy** , Given the relatively small size of the dataset, it is more reasonable to avoid unfreezing a large number of blocks or layers. Limiting the fine-tuning to a small portion of the network—such as 3 blocks or unfreezing 60 to 80 layers—helps prevent overfitting and still provides excellent performance.

The average performance of ResNet50 experiments was weaker than MobileNetV3, due to lighter and more efficient for learning on small data of MobileNetV3 , and considering the training & validation differences across all experiments in ResNet50 **(6.56% difference avg vs 4.32 % difference avg for MobileNetV3),** which gives the impression of an overfitting problem in most cases, with it consuming much more training time, reaching 3-4x of MobileNetV3 with the same hyperparameters. This is mainly due to the complex structure of ResNet, despite its fewer layers (175 vs. 186 for MobileNetV3).

This complex structure also contributed to the overfitting problem in most of cases, with the simple nature of the data, as it is a classification problem with only five classes, not to mention the small size ,even if many layers kept frozen, the complexity of a single layer in ResNet50 gives the impression that it is greedy for larger data to generalize well.

The worst result was in experiment 8 for the ResNet50 model, with a test rate of 73.30%, where 8 blocks were unfreezed, which is a result of a an overfitting , same with experiment 4 (unfreezing

10 blocks led to %98.5 of training accuracy and %86.10 for test set). Therefore, caution should be taken when opening many layers of the complex ResNet50 model to ensure stable performance , as mentioned before, it requires large amounts of training data and memory.

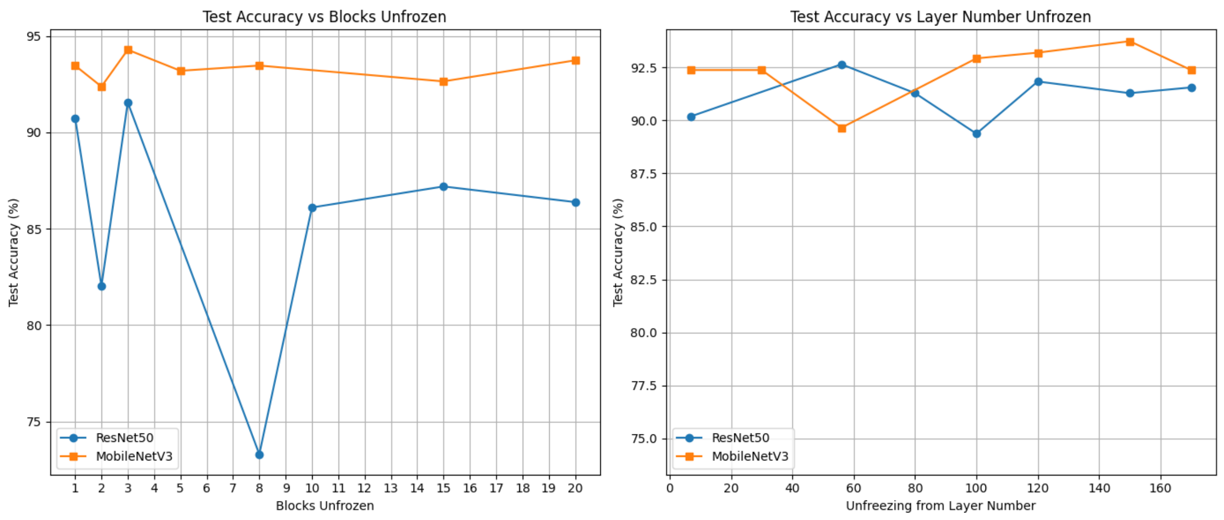

Figure 3 : Comparison of Test Accuracy for ResNet50 and MobileNetV3 Models across Different Block and Layer Freezing Strategies

As a Conclusion, unfreezing blocks is more stable because they are cohesive functional units in the network architecture, despite better flexibility in unfreezing by layers approach, moderate tuning of the number of unfrozen layers or blocks is key to achieving strong and stable performance. At the block level of two to five blocks, or at the layer level, starting with unfrozen layers from layer 100 to 120 will give good generalization performance for both models in general. It is not recommended to go beyond that, as excessive unfreezing will eventually lead to overfitting, and thus worse accuracy on the test data.

– By Ahmad Hudhud