

Optimisation and Evaluation of a MobileNetV2 Neural Network for Plant Pathology Image Classification

Student Name:	Gobinath Gothandapani
Student ID:	2538599
Module (L7):	Application of Machine Learning
Module Code:	ACCA7025
Lecturer:	Dr. Aliyu Sambo
Department:	MSc Data and Artificial Intelligence

TABLE OF CONTENTS

Section	Contents	Page
	Abstract	2
1	Introduction	2
1.1	Background of Neural Networks	2
1.2	Problem Statement	3
1.3	Aim & Objectives	3
2	Related Work	4
2.1	Neural Networks and Optimisation	4
2.2	Recurrent and Convolutional Use Cases	4
2.3	Transfer Learning and MobileNetV2	5
2.4	Critical Comparison of Approaches	5
3	Workflow Methodology	6
3.1	Development Environment	6
3.2	Dataset Loading	6
3.3	Data-Pipeline Optimisation	8
3.4	Augmentation and Input Preprocessing	8
3.5	Model Architecture	8
3.6	Compilation and Initial Training	9
3.7	Fine-tuning Strategy	10
3.8	Evaluation Method	11
4	Results	11
4.1	Initial Feature-Extraction Results	11
4.2	Fine-Tuning Results	12
4.3	Test Evaluation	13
5	Discussion and Recommendations	14
6	Conclusion	15
	References	16

Abstract

Automated image classification can aid in the quick and consistent identification of visible plant health statuses. Nonetheless, the process of classification is hindered by inconsistencies in lighting, orientation, background, and the appearance of symptoms in photographs. This examination constructed and appraised a convolutional neural network classifier employing the scaled dataset from Plant Pathology 2021. An efficient MobileNetV2 architecture was opted for to harmonise predictive capability, model simplicity, and computational performance. The photos were modified to 224×224 pixels and loaded in increments of 32. Random horizontal flipping, rotation, and zoom were implemented to improve the diversity of training data. The implementation of transfer learning occurred in two distinct stages. At the outset, the pretrained MobileNetV2 feature extractor was rendered static as a new classification head was developed for six epochs. Next, the concluding 50 layers were unbound and adjusted for three epochs utilising a diminished learning rate. The methods of Adam optimisation, dropout, early stopping, learning rate reduction, and model checkpointing were employed. The experimental dataset featured 8,691 training photos, 1,862 validation images, and 1,863 test images categorised into 12 distinct groups. Validation accuracy improved from around 71% to 77.82% due to fine-tuning. The definitive model secured a test accuracy of **78.64%** and recorded a test loss of **0.6984**. The research indicates the efficacy of transfer learning in situations of limited training time, yet it is imperative to incorporate class-level measurements and alignment with the defined five-class subset to fortify the conclusive evaluation.

1. Introduction

1.1 Background of Neural Networks

Artificial neural networks learn mappings between inputs and outputs by adjusting weighted connections through forward propagation, loss calculation and backpropagation. Deep networks use multiple layers to learn increasingly abstract representations. In image classification, CNN filters identify local edges, colours and textures before deeper layers combine them into class-relevant structures, reducing dependence on manually engineered features [2], [3].

Plant-pathology images are challenging because the same condition varies with illumination, camera distance, leaf maturity and background, while different conditions may share similar

lesions and discolouration. The Plant Pathology 2021 challenge captures field-like variation and includes individual and combined labels, increasing relevance but also the risk that a model learns background or majority-class correlations [1], [7], [11].

Manual inspection depends on time, expertise and consistent judgement, so an automated classifier is valuable as an initial screening tool rather than a replacement for specialist diagnosis. A custom CNN is transparent but learns all features from random initialisation; a very large architecture may improve accuracy at the cost of memory, training time and overfitting. MobileNetV2 provides a reasoned compromise through compact pretrained feature extraction [2], [9], [10].

1.2 Problem Statement

Manual image inspection can be slow and dependent on specialist experience. Similar visual symptoms may produce inconsistent judgements, particularly when a condition is at an early stage or when multiple symptoms appear in the same image. An automated classifier can provide fast and repeatable initial screening. Nevertheless, a useful model must learn relevant visual evidence rather than memorising backgrounds, image positions or noise.

The assessment also requires the selected model to remain as simple as possible without compromising effectiveness. A small CNN trained from random initialisation is easy to understand, but it may require substantial training to learn all useful image features. Conversely, a very large network may achieve stronger performance but introduce unnecessary computation, memory consumption and overfitting risk. MobileNetV2 was therefore selected as a balanced architecture because it was designed to provide strong visual feature extraction with comparatively low computational requirements.

1.3 Aim and Objectives

The aim of this study was to develop, optimise and critically evaluate a computationally efficient neural network for classifying plant pathology images.[2],[7]

The objectives were:

- 1.To examine artificial neural networks and methods used to optimise their performance.
- 2.To compare appropriate use cases for recurrent and convolutional neural networks.
- 3.To prepare the image dataset through resizing, batching, shuffling and prefetching.
- 4.To use augmentation to improve generalisation.
- 5.To develop a MobileNetV2 transfer-learning classifier.

- 6.To optimise the network through learning-rate adjustment, dropout, callbacks and selective fine-tuning.
- 7.To evaluate the final model using training, validation and independent test performance.
- 8.To identify limitations and propose evidence-based improvements.

2. Related Work

2.1 Neural Networks and Optimisation

A feedforward network transforms inputs through hidden layers into output scores. ReLU enables non-linear representation, while softmax converts final logits into class probabilities. Sparse categorical cross-entropy is appropriate for integer-encoded single labels because it penalises incorrect and overconfident predictions [3], [4].

Adam combines first-moment estimates with adaptive second-moment scaling and is practical for large parameter spaces. Learning-rate control remains essential: a high rate may damage pretrained features, while a very low rate slows convergence. The workflow therefore uses 10^{-3} for the new head and 10^{-5} for fine-tuning [3], [5].

Regularisation limits memorisation of training-specific textures and backgrounds. Dropout reduces co-adaptation, while augmentation introduces plausible orientation and scale changes. Early stopping, learning-rate reduction and checkpointing control training according to validation evidence rather than epoch count alone [4], [8].

2.2 Recurrent and Convolutional Use Cases

Recurrent neural networks maintain a hidden state across ordered observations and are therefore suitable for language, speech, sensor streams and time-series forecasting. Conventional RNNs can experience vanishing or exploding gradients across long sequences, while LSTM networks address this limitation through gated memory cells that regulate storage, updating and forgetting. An RNN would be relevant if the task analysed chronological symptom development or combined images with time-ordered environmental measurements [6].

The present inputs are independent static images, so the decision is driven by spatial relationships within each image rather than information inherited from an earlier observation. CNNs are consequently more appropriate because weight sharing allows a filter to detect a feature in different image locations, and pooling reduces spatial dimensions while retaining salient patterns.

Introducing recurrence would increase complexity without directly addressing lesion shape, colour distribution or texture [2], [6].

2.3 Transfer Learning and MobileNetV2

Transfer learning reuses representations learned from a large source dataset to improve learning on a smaller target task. Earlier pretrained layers tend to represent broadly transferable features such as edges and colour transitions, while deeper layers become more specific to the original task. Freezing the base model initially protects useful generic representations; subsequently unfreezing a limited number of deeper layers enables adaptation without retraining the entire network from random initialisation [2], [5].

MobileNetV2 uses depthwise separable convolutions, inverted residual blocks and linear bottlenecks to reduce operations and parameter count while preserving representational capacity. This makes it suitable where the assessment values simplicity and where future deployment may involve limited hardware. Global average pooling further reduces the classification head compared with Flatten because each feature map is summarised by a single value rather than converted into a large dense vector [2], [10].

Recent studies show that lightweight design still requires critical evaluation. Pal et al. combined MobileNetV2 with residual learning and explainability, while Duhan et al. reported lower performance on the more difficult PlantDoc dataset than on controlled collections. Xu et al. likewise argue that field variation must be central to evaluation [7], [9], [10].

2.4 Critical Comparison of Approaches

Approach	Principal strength	Critical limitation	Relevance
Custom CNN	Simple and fully controlled	Learns all features from random initialisation	Useful baseline but weaker under limited training
RNN/LSTM	Models temporal dependencies	Does not prioritise static spatial structure	Rejected for independent images
ResNet/ConvNeXt	Powerful deep representations	Greater computational and parameter cost	Potential benchmark where resources permit
Vision Transformer	Captures global relationships	Often data- and computation-intensive	Unnecessary for the simplicity objective

Approach	Principal strength	Critical limitation	Relevance
MobileNetV2	Lightweight and transfer-learning ready	May lose performance on difficult field images	Selected balance of efficiency and effectiveness

Table 1 — Critical Comparison of Neural-Network Approaches

The literature does not support declaring one architecture universally superior. Complex models may produce higher accuracy but conflict with the requirement for simplicity. Lightweight models can reduce computation but may lose performance when classes are visually similar or images contain environmental noise. The selection of MobileNetV2 was therefore based on an explicit trade-off: it offered pretrained knowledge, a compact architecture and short training time while remaining capable of selective fine-tuning.

3. Workflow Methodology

3.1 Development Environment

The model was implemented in Python using TensorFlow and Keras within Google Colab. A T4 GPU runtime was selected to accelerate convolution and gradient calculations. The os library was used to construct dataset paths, while Matplotlib was used to display training and validation curves. The root directory was defined as:

```
base_dir = "/content/drive/MyDrive/plant_zip/PlantPathology_Split/PlantPathology_Split"
```

Separate train, valid and test directories were then accessed using os.path.join. This approach is clearer and more portable than repeatedly writing complete folder paths.

3.2 Dataset Loading

Images were loaded using tf.keras.utils.image_dataset_from_directory. This function reads the class names from the subdirectory names and automatically creates integer labels.

The executed output identified:

Partition	Number of images
Training	8,691
Validation	1,862
Test	1,863
Total	12,416

Table-2 Dataset Preparation

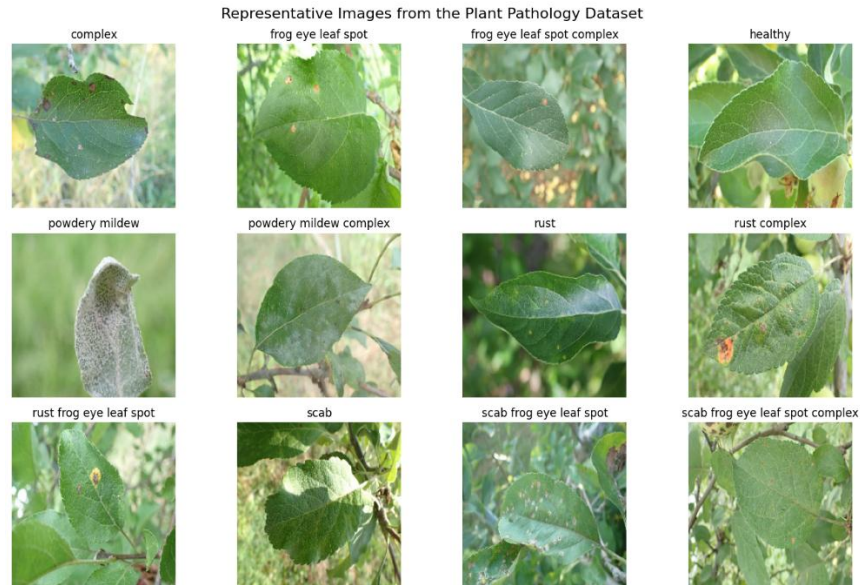


Figure 1 Representative images from the Plant Pathology dataset.

The data loader detected 12 categories:

(complex, frog_eye_leaf_spot, frog_eye_leaf_spot complex, healthy, powdery_mildew, powdery_mildew complex, rust, rust complex, rust frog_eye_leaf_spot, scab, scab frog_eye_leaf_spot, & scab frog_eye_leaf_spot complex.)

This experimental structure was more difficult than a five-class problem because it included individual, combined and complex labels. However, it also represents a deviation from the five-class subset identified in the assessment specification. The result should therefore be interpreted as a broader development experiment rather than evidence of exact compliance with the prescribed subset.

The image dimensions were set to 224×224 pixels:

```
IMG_SIZE = (224, 224)
BATCH_SIZE = 32
```

The reduction from 512×512 to 224×224 matched the standard MobileNetV2 input configuration and reduced processing time and memory consumption. A batch size of 32 allowed the GPU to process several images simultaneously while retaining a reasonable number of gradient updates per epoch.

The training dataset used shuffle=True and a seed of 42. Shuffling reduces the risk that image order affects learning, while the fixed seed improves reproducibility. Validation and test data used shuffle=False, which is important when predictions must later be matched with their true labels.

3.3 Data-Pipeline Optimisation

The three datasets were processed using:

```
AUTOTUNE = tf.data.AUTOTUNE

train_ds = train_ds.prefetch(AUTOTUNE)

valid_ds = valid_ds.prefetch(AUTOTUNE)

test_ds = test_ds.prefetch(AUTOTUNE)
```

Prefetching allows TensorFlow to prepare a future batch while the GPU processes the current batch. It improves utilisation of the available hardware by reducing time spent waiting for data. AUTOTUNE allows TensorFlow to select an appropriate prefetch buffer automatically.

3.4 Augmentation and Input Preprocessing

The model contained three augmentation layers:

```
layers.RandomFlip("horizontal"),

layers.RandomRotation(0.1),

layers.RandomZoom(0.1).
```

Horizontal flipping represented changes in image orientation. Rotation allowed moderate angular variation, while zoom represented differences in camera distance. These transformations were applied only during training. Validation and test images remained unchanged, ensuring that evaluation reflected the original data.

The images were rescaled using:

```
layers.Rescaling(
    1./127.5,
    offset=-1
```

This transformed RGB values from 0–255 to approximately –1 to 1, matching the preprocessing expected by MobileNetV2. Consistent preprocessing is important because the pretrained filters were developed using a particular input distribution.

3.5 Model Architecture

The MobileNetV2 base model was created with:

```
base_model = tf.keras.applications.MobileNetV2(
    input_shape=(224, 224, 3),
    include_top=False,
    weights="imagenet"
)
```

`include_top=False` removed the original ImageNet output layer, while `weights="imagenet"` loaded the pretrained feature extractor. The base model was initially frozen:

```
layer.trainable = False
```

The new classification model consisted of:

1. A 224×224 RGB input layer.
2. Random augmentation layers.
3. MobileNetV2-compatible rescaling.
4. The pretrained MobileNetV2 feature extractor.
5. Global average pooling.
6. Dropout at 0.30.
7. A dense layer containing 128 ReLU neurons.
8. A second dropout layer.
9. A 12-neuron softmax output.

Global average pooling converted the final $7 \times 7 \times 1280$ feature map into a 1,280-element vector. It was preferable to `Flatten` because it substantially reduced the number of parameters in the classification head. The initial model contained approximately 2.42 million parameters, of which only 165,516 were trainable during the frozen stage.

3.6 Compilation and Initial Training

The model was compiled using Adam, sparse categorical cross-entropy and accuracy:

```
model.compile(
    optimizer=tf.keras.optimizers.Adam(
        learning_rate=0.00001
    ),
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)
```

Sparse categorical cross-entropy was appropriate because each image had one integer-encoded label. The initial learning rate of 0.001 enabled the newly created classification head to learn relatively quickly while the pretrained base remained protected.

Three callbacks were applied. Early stopping monitored validation loss and restored the strongest weights if performance stopped improving. ReduceLROnPlateau reduced the learning rate when validation loss reached a plateau. Model checkpointing saved the model associated with the highest validation accuracy.

The initial feature-extraction stage was trained for six epochs. This was a compromise between performance and the available time. The intention was not to maximise the number of epochs but to produce a stable classifier before fine-tuning the pretrained layers.

Parameter	Selected value
Base architecture	MobileNetV2
Pre-trained weights	ImageNet
Input size	$224 \times 224 \times 3$
Batch size	32
Initial learning rate	0.001
Fine-tuning learning rate	0.00001
Initial epochs	6
Fine-tuning epochs	3
Fine-tuned layers	Final 50 layers
Dropout rate	0.30
Optimiser	Adam
Loss function	Sparse categorical cross-entropy
Output activation	Softmax

Table 3 — Model Configuration and Hyperparameters

3.7 Fine-Tuning Strategy

After the initial training stage, MobileNetV2 was unfrozen:

```
base_model.trainable = True
```

Only the final 50 layers were permitted to train:

```
for layer in base_model.layers[:-50]:
```

```
    layer.trainable = False
```

Earlier layers were retained because they contained broadly transferable visual features. Higher layers were adapted because they represented more specialised patterns. This decision aligns with transfer-learning research showing that deeper representations become increasingly task-specific. Batch-normalisation layers were kept frozen to prevent changes to their pretrained moving statistics. The model was recompiled using a much smaller learning rate of 0.00001. This reduced the risk that large gradient updates would destroy useful pretrained knowledge. Fine-tuning was then performed for three epochs.

3.8 Evaluation Method

Accuracy and sparse categorical cross-entropy loss were used in the executed code. Accuracy measures the proportion of images assigned to the correct class, while loss also considers the confidence of the predictions.

For a stronger final evaluation, accuracy should be supplemented with precision, recall, F1-score and a confusion matrix. This is particularly important where categories are imbalanced or visually related. Recent studies commonly use macro precision and macro F1-score to evaluate whether performance is distributed evenly across classes rather than dominated by the largest category.

4. Results

4.1 Initial Feature-Extraction Results

Epoch	Training accuracy	Validation accuracy	Validation loss
1	54.72%	67.40%	1.0630
2	62.48%	70.30%	0.9609
3	65.93%	70.89%	0.9487
4	66.22%	71.27%	0.9471
5	66.76%	71.54%	0.9058
6	66.91%	70.95%	0.9160

Table 4 — Initial Feature-Extraction Results

The initial stage produced a rapid improvement during the first two epochs, after which validation performance improved more gradually. This pattern indicates that the pretrained features were relevant, but the frozen network had limited ability to adapt to the specialised categories.

The training behaviour of the model was evaluated using the accuracy and loss curves presented in Figure 2.

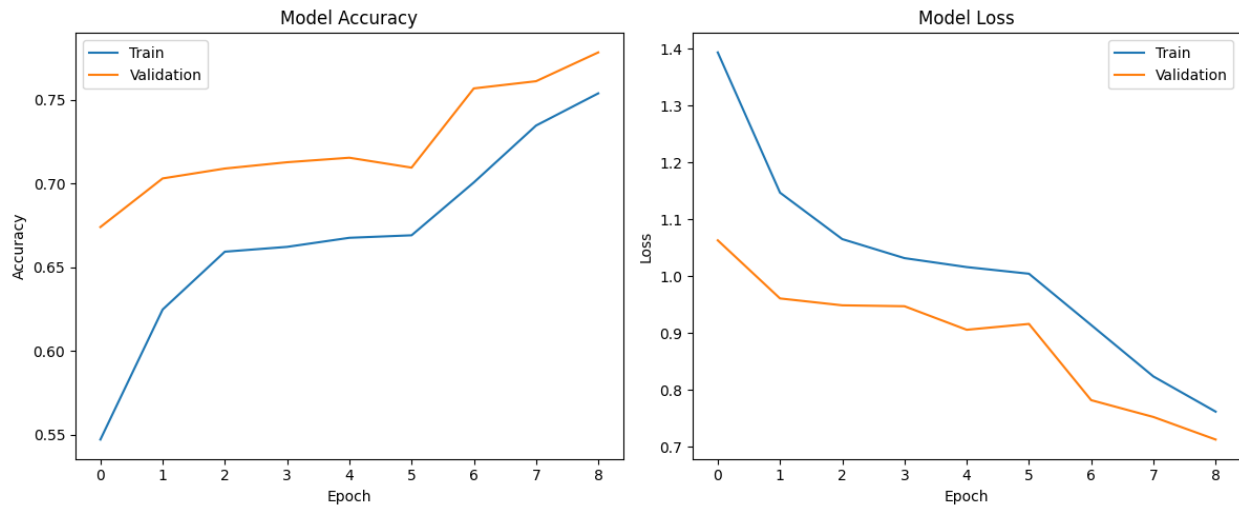


Figure 2 Training and validation accuracy and loss during initial training and fine-tuning.

Figure 2 demonstrates that fine-tuning improved validation accuracy and reduced validation loss. The relatively small difference between training and validation performance indicates that severe overfitting was not observed.

4.2 Fine-Tuning Results

Fine-tuning epoch	Training accuracy	Validation accuracy	Validation loss
1	70.06%	75.67%	0.7820
2	73.46%	76.10%	0.7524
3	75.38%	77.82%	0.7128

Table 5 — Training and Fine-Tuning Performance

Fine-tuning improved validation accuracy from approximately 71% to 77.82%. Validation loss also decreased throughout the fine-tuning stage, demonstrating that the model was becoming both more accurate and more confident.

4.3 Test Evaluation

Metric	Result
Test accuracy	78.64%
Test loss	0.6984

Table 6 — Final Test Performance

The test accuracy was close to, and slightly higher than, the final validation accuracy. This suggests that the model generalised reasonably to unseen images and that there was no evidence of severe overfitting.

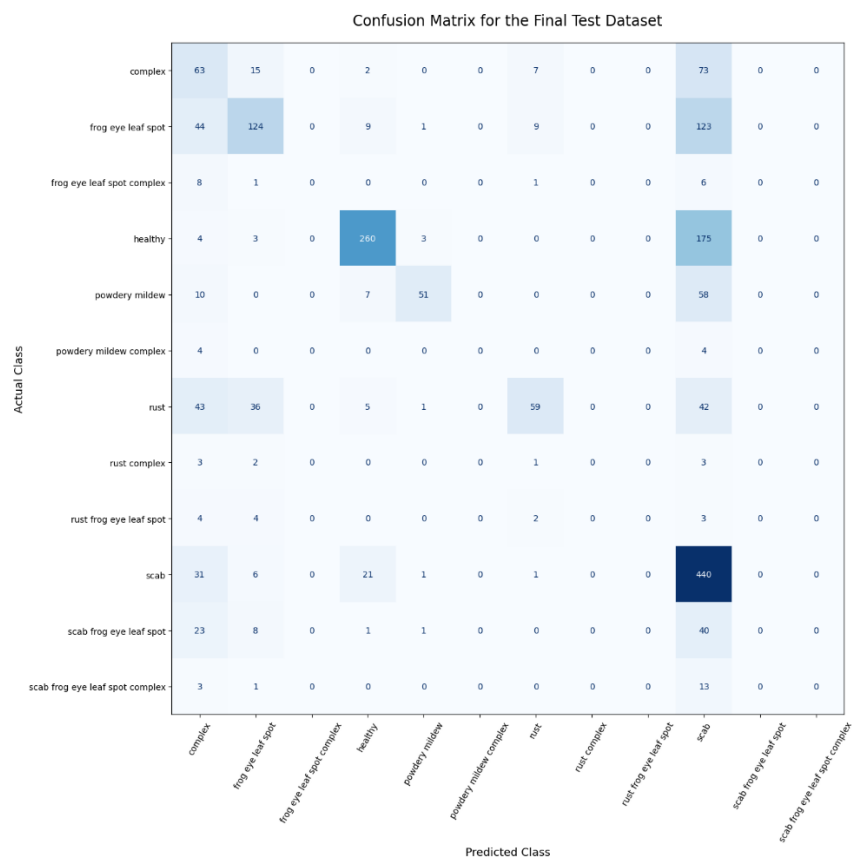


Figure 3 Confusion matrix for the final test dataset

Fig.3 Shows which classes were correctly classified and which classes were confused. This is important because your current accuracy value of **78.64%** does not show individual class performance.

5. Discussion and Recommendations

The results show that transfer learning was effective. The initial frozen model reached approximately 71% validation accuracy after six epochs, while selective fine-tuning increased performance to 77.82%. This improvement demonstrates that generic ImageNet features provided a valuable foundation but that the deeper representations required adaptation to the target image patterns.

Validation accuracy was consistently higher than training accuracy. This does not automatically indicate a methodological problem. During training, augmentation altered the images and dropout temporarily removed neural units, making training predictions more difficult. During validation, augmentation and dropout were inactive. The validation process therefore used stable images and the complete network, which can produce higher accuracy than the training process.

The final 78.64% test accuracy represents a considerable improvement over the earlier basic result of approximately 45%. The improvement can be attributed to pretrained feature extraction, additional training, selective fine-tuning and appropriate input scaling. Nevertheless, it should not be interpreted as proof that the model is ready for deployment. Overall accuracy does not reveal whether every category was recognised consistently.

The presence of combined and complex categories increased the difficulty of the experiment. For example, a class containing symptoms associated with multiple conditions may visually overlap with several single-condition classes. A softmax classifier must still select one category, even where the image contains evidence relevant to more than one label. The original Plant Pathology 2021 dataset explicitly included multi-label cases, illustrating why these distinctions can be difficult.

The result should also be interpreted cautiously when compared with published research. Recent lightweight models sometimes report accuracy above 95%, but many use controlled datasets, extensive augmentation, longer training, larger data volumes or different class definitions. Pal et al., for example, evaluated a specialised residual MobileNet-based architecture on large benchmark datasets and incorporated a more extensive training and explanation workflow. A recent ConvNeXt-based study also showed substantial performance differences between controlled and real-world data, confirming that accuracy values are not directly comparable across experimental conditions.

The principal assessment risk is the dataset configuration. The specification requires five categories, but the implemented loader identified 12. For full practical compliance, the final dataset should contain only the five required folders. The current result may be retained as evidence of experimentation, but the final submitted model should ideally be rerun using the prescribed subset.

The following improvements are recommended:

1. **Five-class compliance:** Retain only the required categories and update the final metrics.
2. **Class distribution analysis:** Report the number of images per category and identify imbalance.
3. **Classification report:** Calculate precision, recall and F1-score for each category.
4. **Confusion matrix:** Identify the categories most frequently confused.
5. **Baseline comparison:** Compare the original custom CNN with frozen and fine-tuned MobileNetV2.
6. **Longer controlled training:** Permit additional fine-tuning epochs while retaining early stopping.
7. **Repeated experimentation:** Train using more than one random seed to test result stability.
8. **Interpretability:** Use Grad-CAM to confirm that predictions are based on relevant image areas rather than backgrounds.
9. **External evaluation:** Test the model on independently captured images to measure real-world generalisation.

These recommendations preserve the requirement for simplicity. A larger architecture should only be introduced where it produces a clear and measurable improvement over MobileNetV2.

6. Conclusion

This study developed and evaluated a MobileNetV2 transfer-learning model for plant pathology image classification. RNN and LSTM architectures were considered conceptually but rejected because the task involved independent static images rather than ordered sequences. A CNN was more appropriate because it could learn spatial features including colour variation, texture and symptom structure.

The implemented workflow included image resizing, batching, shuffling, prefetching, augmentation, MobileNetV2 preprocessing, global average pooling, dropout, Adam optimisation, early stopping, learning-rate reduction and model checkpointing. Training was divided into frozen

feature extraction and selective fine-tuning. This strategy increased validation accuracy from approximately 71% to 77.82%.

The final model achieved **78.64% test accuracy** and a **0.6984 test loss**. The close validation and test results indicate reasonable generalisation, although class-level metrics are needed for a complete evaluation. MobileNetV2 achieved a defensible balance between simplicity, computational efficiency and effectiveness. The most important remaining requirement is to align the final practical experiment with the specified five-class subset and support the accuracy result with precision, recall, F1-score and confusion-matrix evidence.

References

- [1] S. Belongie, Q. Wang, N. Snavely and A. Khan, “The Plant Pathology 2021 Challenge dataset to classify foliar disease of apples,” Plant Pathology 2021 Challenge—FGVC8, 2021.
- [2] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.
- [3] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in Proc. International Conference on Learning Representations, 2015.
- [4] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” Journal of Machine Learning Research, vol. 15, no. 56, pp. 1929–1958, 2014.
- [5] J. Yosinski, J. Clune, Y. Bengio and H. Lipson, “How transferable are features in deep neural networks?” in Advances in Neural Information Processing Systems, vol. 27, 2014.
- [6] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
- [7] M. Xu, J.-E. Park, J. Lee, J. Yang and S. Yoon, “Plant disease recognition datasets in the age of deep learning: Challenges and opportunities,” Frontiers in Plant Science, vol. 15, Art. no. 1452551, 2024, doi: 10.3389/fpls.2024.1452551.
- [8] A. Y. Ashurov et al., “Enhancing plant disease detection through deep learning: A depthwise CNN with squeeze and excitation integration and residual skip connections,” Frontiers in Plant Science, vol. 15, Art. no. 1505857, 2025, doi: 10.3389/fpls.2024.1505857.

- [9] C. Pal, S. Karmakar, I. Mukherjee and P. P. Chakrabarti, “A lightweight and explainable CNN model for empowering plant disease diagnosis,” *Scientific Reports*, vol. 15, Art. no. 30720, 2025, doi: 10.1038/s41598-025-94083-1.
- [10] R. Thapa, K. Zhang, N. Snavely, S. Belongie and A. Khan, “The Plant Pathology Challenge 2020 data set to classify foliar disease of apples,” *Applications in Plant Sciences*, vol. 8, no. 9, Art. no. e11390, 2020, doi: 10.1002/aps3.11390.
- [11] R. R. Selvaraju et al., “Grad-CAM: Visual explanations from deep networks via gradient-based localization,” in *Proc. IEEE International Conference on Computer Vision*, 2017, pp. 618–626.