Green Intelligent Systems and Applications

Research Article
Volume 6(1), 2026, 174−194
https://doi.org/10.53623/gisa.v6i1.1239

Classification of Leaf Diseases in Guava Plants Based on Images Using the MobileNetV3 Model

Rikky*, Ery Hartati

Faculty of Computer Science and Engineering, Universitas Multi Data Palembang, South Sumatra, Indonesia

*Correspondence: rikky25@mhs.mdp.ac.id

SUBMITTED: 8 June 2026; REVISED: 25 June2026; ACCEPTED: 29 June 2026

ABSTRACT: Guava cultivation is often threatened by leaf diseases that disrupted plant growth and reduced agricultural productivity. Early and accurate disease identification was crucial but relied heavily on slow and subjective visual inspections by human experts. This study proposed an automated, accurate, and efficient solution by comparing lightweight deep learning models. A total of 656 augmented guava leaf images representing four classes (Algal Leaf Spot, Insects Eaten, Red Rust, and Healthy Leaf) were evaluated. Using a transfer learning approach, the hyperparameters were systematically tuned for MobileNetV3-Small as the proposed model and compared with MobileNetV2 as the baseline architecture. The experimental results demonstrated that MobileNetV3-Small achieved a superior test accuracy of 91.00%, outperforming MobileNetV2, which achieved 87.00%. The integration of Squeeze-and-Excitation (SE) modules and the h-swish activation function in MobileNetV3-Small significantly improved the identification of subtle visual symptoms, particularly for the Healthy Leaf and Insects Eaten classes. However, MobileNetV2 maintained a slight advantage in real-time processing speed (54.86 FPS versus 50.30 FPS) because of memory-bound bottlenecks associated with the SE modules. Overall, MobileNetV3-Small provided superior diagnostic accuracy, whereas MobileNetV2 remained a highly viable option for latency-critical deployment on low-end devices.

KEYWORDS: Deep Learning; guava leaf disease; image classification; MobileNetV2; MobileNetV3-Small

1. Introduction

Agriculture is a primary human activity involving the cultivation of plants to meet food needs. One of the key sectors within agriculture is horticulture, which encompasses fruit crops, vegetables, ornamental plants, and medicinal plants. One tropical fruit commodity with high economic value in Indonesia is guava (*Psidium guajava* L.) [1]. In addition to its popular flavor, guava is rich in vitamin C and dietary fiber, resulting in continuously increasing demand. Data from the Central Statistics Agency (BPS) showed that guava production in Indonesia reached 418,124.5 tons in 2024 [2]. This high production level indicated strong market demand as well as the importance of maintaining productivity and harvest quality.

Nevertheless, guava cultivation continued to face various challenges, one of which was leaf disease caused by fungi, bacteria, or pests [3]. Leaf diseases disrupted photosynthesis,

inhibited plant growth, reduced productivity, and could even cause plant death. These impacts resulted not only in economic losses due to reduced crop yields but also in social consequences and implications for food security. Previous research indicated that economic and environmental factors significantly influenced agricultural productivity and food security in rural communities, particularly when farmers had limited access to capital and agricultural technology [4]. Additionally, disruptions in agricultural supply chains and commodity production increased food insecurity among small-scale farmers [5]. Therefore, effective solutions were needed to help farmers detect plant diseases quickly and accurately to minimize the risk of crop failure.

Currently, the identification of diseases in guava leaves was generally performed visually by farmers and agricultural experts [6]. This conventional method required experience and specialized expertise and was relatively time-consuming, especially in areas with a shortage of experts [6]. Furthermore, the manual identification process was subjective, which could lead to misdiagnosis. Misidentification could result in inappropriate treatment, wasted resources, and further deterioration of plant health.

Advances in artificial intelligence (AI), particularly deep learning, have offered promising solutions to these challenges. Deep learning is a branch of machine learning capable of processing large volumes of data and achieving high accuracy in image pattern recognition [7]. One of the most widely used approaches for image classification is the Convolutional Neural Network (CNN), which performs automatic and effective feature extraction. Various studies have demonstrated the effectiveness of CNNs in classifying plant diseases. A study conducted by Syahputra used the MobileNetV2 architecture and achieved an accuracy of 99.72% in classifying honey guava leaf pests [8]. Another study by Fadlan et al. achieved an accuracy of 91.6% in classifying honey guava leaf and fruit diseases using MobileNetV2 [1]. Furthermore, research on corn leaf disease classification showed that MobileNetV3-Small achieved an accuracy of 99.5%, while MobileNetV3-Large achieved 99.0% [9]. These findings indicated that MobileNetV3 possessed considerable potential for application in various plant disease classification tasks.

MobileNetV3 is an evolution of MobileNetV1 and MobileNetV2 that was designed to achieve higher accuracy while requiring fewer computational resources. These characteristics make MobileNetV3 suitable for deployment on resource-constrained devices such as smartphones and Internet of Things (IoT) devices. Previous research also showed that MobileNetV3 achieved an accuracy of up to 99.84% in grapevine leaf disease identification while maintaining a relatively lightweight model [10].

Although various studies have successfully applied CNNs to plant disease classification and reported high accuracy, research specifically implementing and comparing the performance of MobileNetV3 and MobileNetV2 for guava leaf disease classification remained very limited [1,8]. Most previous studies focused on a single architecture without conducting comparative analyses of newer and more efficient models. MobileNetV3 was developed as an improvement over MobileNetV2 by incorporating several architectural enhancements aimed at improving classification accuracy while reducing computational complexity. This research gap highlighted the need to evaluate the extent to which MobileNetV3 could provide better performance and computational efficiency than MobileNetV2 for guava leaf disease classification.

Based on this research gap, this study proposes the use of the MobileNetV3-Small architecture with the Adam optimizer for guava leaf disease classification and compares its performance with MobileNetV2 as the baseline model. MobileNetV3-Small was selected because it can produce lightweight models while maintaining high classification performance, making it suitable for deployment on resource-constrained devices [11]. This choice was further supported by recent comparative studies benchmarking lightweight architectures, such as EfficientNet-Lite, ShuffleNet, and MobileViT, for plant disease classification. These studies showed that MobileNetV2 and MobileNetV3-Small remained among the most consistently competitive and widely adopted backbone models for transfer-learning-based agricultural image classification on resource-constrained devices, whereas broader evaluations of newer architectures remain an area for future research [12, 13]. Additionally, the Adam optimizer was selected because it accelerated convergence and improved training stability [14]. Through this study, a guava leaf disease classification model is expected to be developed that not only achieves high classification accuracy but also maintains computational efficiency. The findings are expected to support the application of artificial intelligence in agriculture, particularly by enabling farmers to detect plant diseases at an early stage more quickly and accurately.

2. Materials and Methods

2.1. Deep learning.

Deep learning is a type of machine learning that performs exceptionally well on large-scale and unstructured data, such as images and text, while effectively addressing complex problems that are difficult to solve using classical methods. Consequently, deep learning enables computational models to learn hierarchical feature representations from data across multiple levels [15]. An illustration of deep learning is shown in Figure 1 [16].

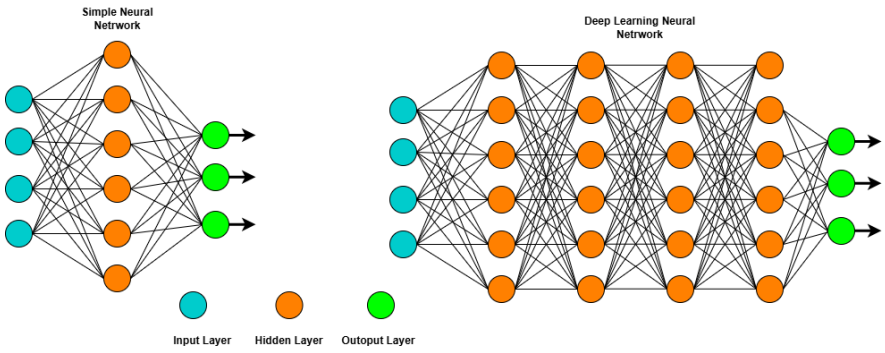

Figure 1. Architecture of Deep Learning

As shown in Figure 1, the learning process in deep learning is performed through artificial neural networks composed of multiple hidden layers [17]. Each layer extracts increasingly complex data representations, ranging from simple features such as edges and colors in images to more abstract features that characterize specific object classes. Through this hierarchical learning process, the model automatically learns from large and complex datasets and progressively improves its performance as the amount of training data and the number of training iterations increase. The primary advantage of deep learning is its ability to learn highly representative features from large-scale datasets with high predictive accuracy. However, it requires substantial computational resources as well as careful hyperparameter tuning and regularization to prevent overfitting. Common deep learning architectures include the

Convolutional Neural Network (CNN), which is widely used for image analysis, and the Recurrent Neural Network (RNN), which is designed for sequential data such as text and speech, including its advanced variants, the Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) networks [18]. Compared with traditional machine learning approaches, which rely on manually engineered features such as color histograms, texture descriptors, and shape statistics, deep learning automatically learns hierarchical feature representations directly from raw pixel data. This automated feature extraction is particularly advantageous for plant disease classification because disease symptoms vary considerably in shape, color intensity, and texture, making them difficult to capture using fixed, manually designed feature descriptors. Consequently, deep learning models have consistently outperformed conventional machine learning methods based on handcrafted features in plant disease classification tasks [19].

2.2. Convolutional Neural Network (CNN).

CNN) is a type of artificial neural network architecture specifically designed to process grid-structured data, particularly images, and has been widely used in computer vision tasks such as image classification, object detection, and image segmentation [20]. CNN operates through feature extraction and classification stages by applying convolution operations with small filters (kernels) to automatically learn spatial and hierarchical features from images. Its architecture generally consists of convolutional layers, activation layers such as Rectified Linear Unit (ReLU), pooling layers for dimensionality reduction and overfitting prevention, and fully connected layers that generate the final prediction for classification, detection, or segmentation tasks. The CNN architecture is illustrated in Figure 2. In the specific context of leaf disease classification, the early convolutional layers learn low-level features such as edges, color transitions, and texture patterns, whereas the deeper layers combine these low-level features into higher-level representations corresponding to disease-related visual characteristics, including lesion shape, discoloration patterns, and surface texture irregularities. This hierarchical learning process is broadly analogous to the way human experts visually examine leaves by identifying progressively more distinctive disease symptoms [21, 22].

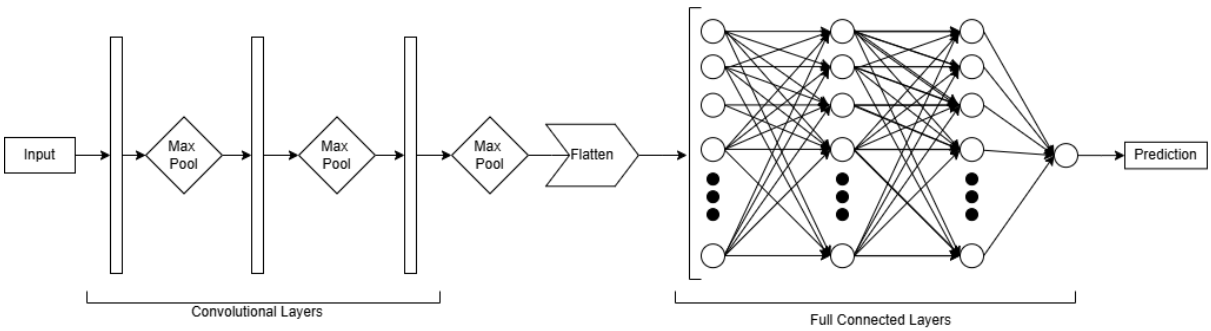

Figure 2. Architecture CNN

2.3. MobileNetV2.

MobileNetV2 is a convolutional neural network architecture introduced by Sandler et al. in 2018 as an improvement over MobileNetV1. The architecture was designed using a sequence of inverted residual blocks that operate in multiple stages. Each block begins with a 1 × 1 expansion convolution that increases the number of channels from a low-dimensional space to a higher-dimensional space, followed by a 3 × 3 depthwise convolution that performs spatial

filtering independently on each channel. The block ends with a 1 × 1 projection convolution that reduces the channel dimension back to the original bottleneck size. When the input and output share the same spatial resolution and channel dimensions, a residual (shortcut) connection is incorporated to facilitate information flow and improve gradient propagation during training [23]. The architecture of MobileNetV2 is presented in Table 1.

Table 1. Architecture MobileNetV2.

Input	Operator
224 × 224 × 3	Conv2d
112 × 112 × 32	Bottleneck
112 × 112 × 16	Bottleneck
56 × 56 × 24	Bottleneck
28 × 28 × 32	Bottleneck
14 × 14 × 64	Bottleneck
14 × 14 × 96	Bottleneck
7 × 7 × 160	Bottleneck
7 × 7 × 320	Conv2d 1 × 1
7 × 7 × 1280	Avgpool 7 × 7
1 × 1 × 1280	Conv2d 1 × 1

The principal characteristic of MobileNetV2 is its linear bottleneck design. In this architecture, nonlinear activation functions such as ReLU are omitted in the final projection layer to preserve important information within the low-dimensional feature space. Consequently, the model is able to learn rich feature representations while maintaining low computational cost. This design also makes MobileNetV2 an effective feature extractor for transfer learning and hybrid architectures, while remaining well suited for deployment on resource-constrained devices [24]. A comprehensive review by Ekmekyapar and Taşcı [25] further demonstrated the effectiveness of MobileNetV2 across a wide range of real-world applications. Their study also highlighted several successful adaptation strategies, including the integration of lightweight attention modules, model quantization, knowledge distillation, and the replacement of standard building blocks with more computationally efficient variants. These adaptations have enabled MobileNetV2 to remain a competitive architecture despite the introduction of newer MobileNet generations [25]. An illustration of the MobileNetV2 architecture is shown in Figure 3.

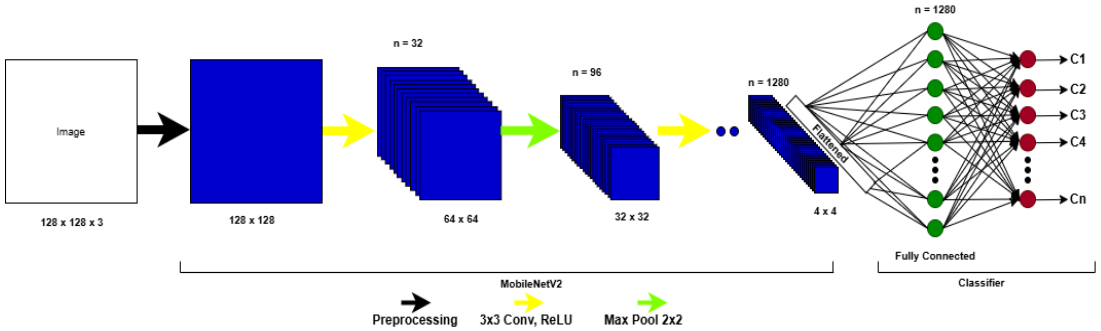

Figure 3. Illustrate of Architecture MobileNetV2.

2.4. MobileNetV3-Small.

MobileNetV3 is a convolutional neural network architecture developed by Google to address the computational constraints of resource-limited devices. Its primary objective is to provide a lightweight, fast, and efficient deep learning model that can be deployed effectively across a wide range of mobile and embedded platforms. MobileNetV3 was introduced in two main

variants, namely MobileNetV3-Large and MobileNetV3-Small, to accommodate different application scenarios. The MobileNetV3-Large variant was designed for tasks requiring greater computational capacity and higher predictive performance, whereas the MobileNetV3-Small variant was optimized for environments with more stringent computational and memory constraints. Both variants support a variety of computer vision tasks, including image classification, object detection, and image segmentation [23]. The architecture of MobileNetV3-Small is illustrated in Figure 4 [26].

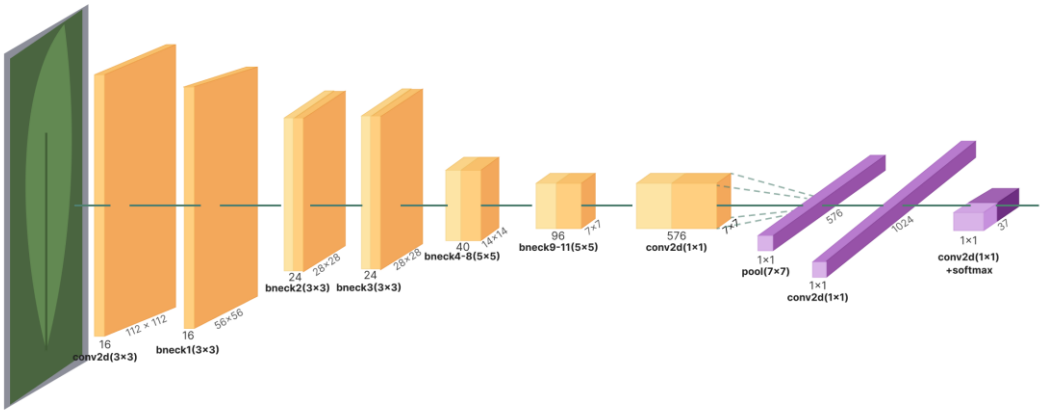

Figure 4. Illustrated of Architecture MobileNetV3-Small

MobileNetV3 is an advancement of MobileNetV2 that integrates several efficiency mechanisms, including inverted residual blocks, depthwise separable convolutions, Squeeze-and-Excitation (SE) modules, and the hard-swish (h-swish) activation function [27]. Depthwise separable convolution divides the convolution operation into depthwise and pointwise stages, significantly reducing the number of parameters and computational cost. In each bottleneck block, the first 1 × 1 convolution performs channel expansion, followed by a 3 × 3 depthwise convolution for spatial feature extraction and a final 1 × 1 convolution for channel compression. MobileNetV3-Large repeats the bottleneck structure 15 times, whereas MobileNetV3-Small uses 11 lighter bottleneck blocks to improve computational efficiency.

In addition, the Squeeze-and-Excitation (SE) module adaptively adjusts channel weights through global average pooling and nonlinear activation, enabling the model to emphasize important features. The h-swish activation function is used as a computationally efficient approximation of the swish function, providing strong nonlinear modeling capability while remaining lightweight. The combination of these architectural components enables both MobileNetV3 variants to achieve superior performance, with MobileNetV3-Large offering higher classification accuracy and MobileNetV3-Small being well suited for deployment on devices with limited computational resources. The architecture of MobileNetV3-Small is presented in Table 2. In the context of plant disease classification, the SE module enables the network to assign greater importance to feature channels that capture disease-specific color and texture characteristics while suppressing less relevant background information, allowing the model to focus on diagnostically important regions of the leaf [31]. The original SE-Network study reported that this channel reweighting introduced only a small increase in FLOPs and model parameters relative to the improvement in classification accuracy. However, it also

introduced additional runtime overhead from the pooling and fully connected operations that was not fully reflected by FLOP measurements alone [28].

Table 2. Architecture MobileNetV3-Small.

Input	Operator	exp size	*#out*	SE	NL	s
224^2 x 3	conv2d, 3x3	-	16	-	HS	2
112^2 x 16	bnec,3x3	16	16	✓	RE	2
56^2 x 16	bnec,3x3	72	24	-	RE	2
28^2 x 24	bnec,3x3	88	24	-	RE	1
28^2 x 24	bnec,5x5	96	40	✓	HS	2
14^2 x 40	bnec,5x5	240	40	✓	HS	1
14^2 x 40	bnec,5x5	240	40	✓	HS	1
14^2 x 40	bnec,5x5	120	48	✓	HS	1
14^2 x 48	bnec,5x5	144	48	✓	HS	1
14^2 x 48	bnec,5x5	288	96	✓	HS	2
7^2 x 96	bnec,5x5	576	96	✓	HS	1
7^2 x 96	bnec,5x5	576	96	✓	HS	1
7^2 x 96	conv2d, 1x1	-	576	✓	HS	1
7^2 x 576	pool, 7x7	-	-	-	-	1
1^2 x 576	conv2d 1x1, NBN	-	1280	-	HS	1
1^2 x 1280	conv2d 1x1, NBN	-	k	-	-	1

In the MobileNetV3 specifications, the “SE” column indicates whether the input block uses Squeeze-and-Excitation or not. The “NL” column describes the type of activation function used, where HS stands for h-swish and RE stands for ReLU. Meanwhile, the notation NBN in the “Operator” column means that the block does not use batch normalization. The “s” column indicates the stride used in the block, and #out represents the number of filters used in the architecture.

2.5. Transfer learning.

Transfer Learning is a technique in machine learning in which a model no longer needs to learn from scratch, as it can utilize knowledge and experience from related sources for new tasks. This approach is highly useful in limited domains, where training a model from the beginning becomes inefficient. This definition is in line with Matt’s statement cited in [29]. According to Matt, as cited by Asmaul Hosna [29] tells about Transfer Learning is defined as a category in machine learning that reuses existing models to solve current challenges. Matt also stated that transfer learning is a technique for training models simultaneously, where existing training data concepts are utilized to improve model performance so that solutions do not need to be developed from the ground up.

2.6. Optimizer.

An optimizer is an optimization algorithm that enables deep neural networks to learn. The effectiveness of the learning process highly depends on the optimizer used during the training process. Its primary objective is to indirectly improve the model’s performance metrics, which are difficult to optimize directly [30]. An optimizer works by iteratively minimizing the loss function. Adam was selected over alternatives such as SGD and RMSprop because it combines the benefits of both: it uses a per-parameter adaptive learning rate similar to RMSprop while also incorporating momentum-like updates based on the moving average of past gradients, which generally allows it to converge faster and more reliably than vanilla SGD on relatively small, fine-tuning-style datasets, without requiring the extensive manual learning-rate

scheduling that SGD typically needs. This is consistent with comparative studies reporting that Adam consistently matches or outperforms RMSprop and plain SGD in validation accuracy on small- to medium-sized image-classification datasets [31].

2.7. Dataset collection.

In this study, the dataset used consists of guava plant leaves. The images of the guava leaves were obtained from a publicly available dataset on Kaggle titled “Dataset of Guava Leaf Disease in Bangladesh”. The original dataset size is around 1.9 GB before augmentation, and this is categorized into four classes: algal_leaf_spot, insects_eatten, red_rust, and healthy_leaf. More detailed information can be seen in Figure 4.

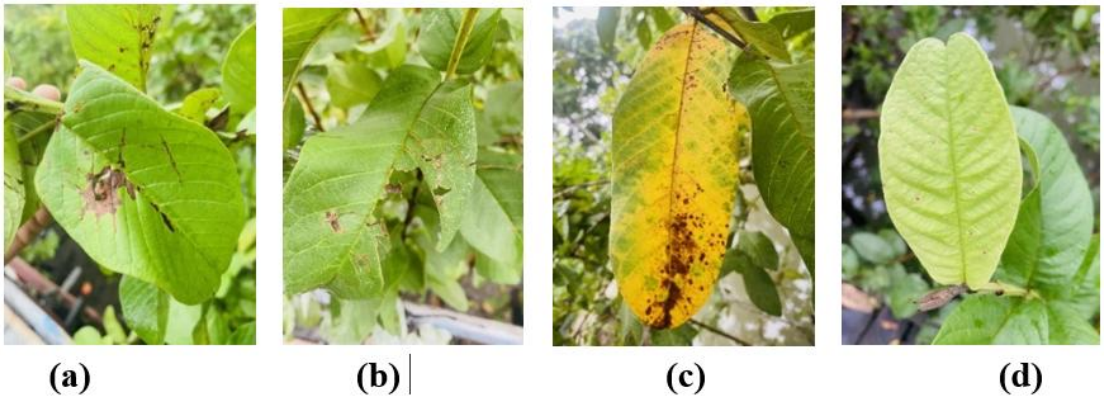

Figure 5. Guava leaf disease dataset samples: (a) Algal Leaf Spot – This is the first type of disease found on guava leaves, caused by algal parasites, characterized by small brown spots on the upper surface of the leaves. (b) Insect Eaten - This is a type of disease on guava leaves caused by insects, characterized by holes in the leaves and torn edges. (c) Red Rust - This is a type of disease on guava leaves characterized by reddish spots resembling rust on the upper surface of the leaves. (d) Healthy Leaf - This is a type of leaf that appears fresh, green, and healthy.

2.8. Research design.

This study designed a guava leaf disease classification system using the MobileNetV2 and MobileNetV3-Small architectures with a transfer learning approach. The system design began with a data preprocessing stage to standardize image quality. This stage included resizing the images to 224 × 224 pixels and normalizing the pixel values to meet the input requirements of the MobileNet architectures and the transfer learning process. The preprocessed dataset was then divided into three subsets: 80% for training, 10% for validation, and 10% for testing. This data split was applied consistently across all experimental scenarios to ensure a fair comparison between the models.

In this study, two training scenarios were designed: MobileNetV2 served as the baseline model, whereas MobileNetV3-Small was used as the proposed model. Both architectures were trained using a transfer learning approach with pretrained weights. In each model, the initial convolutional layers were frozen and used as automatic feature extractors, whereas the final classification layers were modified and fine-tuned using the training data. All training scenarios were conducted using the same configuration parameters, including the dataset, preprocessing scheme, number of epochs, optimizer, and batch size, ensuring that any differences in performance reflected the characteristics of each architecture objectively.

After training, each model was evaluated using an independent test set to measure its generalization capability. Classification performance was assessed using a confusion matrix

and the derived metrics of precision, recall, and F1-score to analyze classification accuracy and misclassification among the guava leaf disease classes. In addition to classification performance, efficiency metrics were also evaluated to compare MobileNetV2 and MobileNetV3-Small, including the number of model parameters, inference time, and computational requirements. The overall system design is illustrated in Figure 6. Specifically, the workflow consisted of six stages: (1) preprocessing, in which raw leaf images were resized, normalized, and augmented; (2) transfer learning, in which ImageNet-pretrained MobileNetV2 and MobileNetV3-Small backbones were loaded and their convolutional base layers were frozen; (3) training, in which the unfrozen classification head was fine-tuned on the training set using the Adam optimizer; (4) validation, in which performance on the validation set was monitored after each epoch to guide hyperparameter selection and detect overfitting; (5) testing, in which the final trained model was evaluated on the independent test set; and (6) evaluation, in which confusion-matrix-based metrics and efficiency metrics were computed and compared between the two architectures.

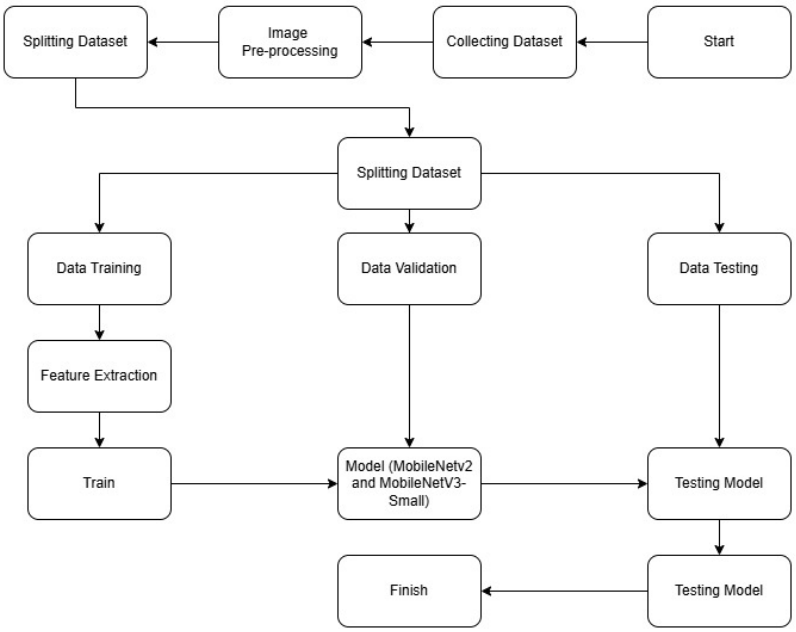

Figure 6. Diagram workflow.

2.9. Parameter configuration.

Hyperparameters are parameters defined before the training process and function to control the learning mechanism of a deep learning model. Proper hyperparameter configuration plays an important role in maintaining training stability, accelerating convergence, and improving the performance and generalization capability of the model. Various studies have shown that adjustments to the learning rate, batch size, number of epochs, and optimizer significantly influence the performance of Convolutional Neural Networks (CNN) in image classification tasks [32]. To ensure that the performance comparison between MobileNetV2 and MobileNetV3 is conducted fairly and objectively, both models were trained using the same hyperparameter configuration. The hyperparameters used include the learning rate, batch size, number of epochs, and optimizer. The values of each hyperparameter used during the training process are presented in Table 3. These values were not chosen arbitrarily: a relatively low learning rate was selected because the convolutional base layers are pretrained on ImageNet

and only the classification head is fine-tuned, so a small learning rate helps avoid disrupting the already-useful pretrained feature representations during the early epochs of training. The batch size was selected to balance gradient-estimate stability against the memory constraints of the available hardware and the limited size of the dataset, while the number of epochs was set high enough to allow the validation accuracy to plateau, with the best-performing epoch retained, so as to avoid both under-training and excessive overfitting. This general approach to hyperparameter selection for Adam-optimized transfer learning is consistent with prior hyperparameter-tuning studies on similarly sized image-classification tasks [26].

Table 3. Configuration hyperparameter.

No	Hyperparameter	Configuration Values
1	Learning rate	0.001 and 0.0001
2	Batch size	16 and 32
3	Epoch	40 and 80
4	Optimizer	Adam

3. Results and Discussion

3.1. Dataset and preprocessing overview.

The dataset used in this study consisted of 656 guava leaf images evenly distributed across four classes: Algal Leaf Spot, Insects Eaten, Red Rust, and Healthy Leaf, with 164 images per class. This balanced distribution was achieved through data augmentation applied to the original Kaggle dataset entitled *Dataset of Guava Leaf Disease in Bangladesh*. Augmentation was performed on the training data using the Keras ImageDataGenerator with a rotation range of 30°, horizontal flipping, a zoom range of 0.2, a shear range of 0.1, and width and height shift ranges of 0.1. These augmentation techniques increased data diversity and reduced the risk of overfitting, thereby improving the model's generalization capability [33]. All images were resized to 224 × 224 pixels and normalized before training to satisfy the input requirements of both MobileNetV2 and MobileNetV3-Small under the transfer learning framework. Following preprocessing and resizing, the dataset size decreased from approximately 1.9 GB to 14.54 MB because the original high-resolution images were converted to a smaller, standardized resolution. The dataset was then divided into training (80%, 524 images), validation (10%, 66 images), and testing (10%, 66 images) subsets. This partitioning strategy was consistently applied across all experiments to ensure a fair and objective comparison between the two models. The use of geometric and photometric augmentation on a relatively small dataset is a widely adopted regularization strategy that improves model robustness and generalization when training deep learning models with limited data [33].

3.2. Test scenarios and overall performance analysis.

A systematic hyperparameter tuning process was conducted to determine the optimal training configuration for each model. Eight configurations were evaluated for each architecture by combining two learning rates (0.001 and 0.0001), two batch sizes (16 and 32), and two epoch settings (40 and 80), while using the Adam optimizer in all experiments. The complete tuning results for MobileNetV2 is presented in Tables 4. MobileNetV2 achieved its highest accuracy of 87.00% in Scenario 3 using 40 epochs, a batch size of 32, a learning rate of 0.001, and the Adam optimizer. Overall, the learning rate of 0.001 consistently produced better performance than 0.0001, suggesting that the larger learning rate enabled more effective optimization during

fine-tuning on the relatively small dataset. Increasing the number of training epochs from 40 to 80 did not consistently improve performance. For example, Scenario 7 (80 epochs, batch size 32, learning rate 0.001) achieved only 76.00% accuracy, which was substantially lower than Scenario 3. This result suggested that prolonged training led to overfitting, reducing the model's ability to generalize to unseen data.

Table 4. Hyperparameter tuning result for MobileNetV2.

Scenario	Epoch	Batch Size	Learning Rate	Optimizer	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
S1	40	16	0.001	Adam	81.00%	82.00%	81.00%	80.00%
S2	40	16	0.0001	Adam	75.00%	75.00%	75.00%	75.00%
S3	40	32	0.001	Adam	87.00%	87.00%	87.00%	87.00%
S4	40	32	0.0001	Adam	78.00%	79.00%	78.00%	78.00%
S5	80	16	0.001	Adam	82.00%	82.00%	82.00%	82.00%
S6	80	16	0.0001	Adam	83.00%	82.00%	82.00%	82.00%
S7	80	32	0.001	Adam	76.00%	76.00%	76.00%	76.00%
S8	80	32	0.0001	Adam	76.00%	75.00%	75.00%	75.00%

As shown in Table 5, MobileNetV3-Small achieved its highest accuracy of 91.00% in Scenario 13 using 80 epochs, a batch size of 16, a learning rate of 0.001, and the Adam optimizer. Similar to MobileNetV2, the higher learning rate consistently outperformed 0.0001 across all configurations, whereas the lower learning rate resulted in slower convergence and inferior classification performance. Unlike MobileNetV2, MobileNetV3-Small achieved its best performance after 80 training epochs rather than 40. This finding suggested that the more sophisticated architecture, which incorporates Squeeze-and-Excitation modules and the h-swish activation function, required additional training iterations to fully optimize its parameters. Consequently, MobileNetV3-Small produced the highest overall classification accuracy despite requiring a longer training process.

Table 5. Hyperparameter tuning result for MobileNetV3-Small.

Scenario	Epoch	Batch Size	Learning Rate	Optimizer	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
S9	40	16	0.001	Adam	88.00%	90.00%	88.00%	88.00%
S10	40	16	0.0001	Adam	76.00%	78.00%	76.00%	76.00%
S11	40	32	0.001	Adam	85.00%	85.00%	85.00%	85.00%
S12	40	32	0.0001	Adam	75.00%	76.00%	75.00%	75.00%
S13	80	16	0.001	Adam	91.00%	92.00%	91.00%	91.00%
S14	80	16	0.0001	Adam	79.00%	81.00%	79.00%	80.00%
S15	80	32	0.001	Adam	84.00%	86.00%	84.00%	84.00%
S16	80	32	0.0001	Adam	71.00%	71.00%	71.00%	70.00%

3.3. MobileNetV2 Model performance analysis.

This section presents the performance analysis of the MobileNetV2 model under the optimal training scenario (Scenario 3), configured with 40 Epochs, a Batch Size of 32, a Learning Rate of 0.001, and the Adam Optimizer. The evaluation is grounded on the training performance graphs and the confusion matrix applied to the testing dataset, which ultimately yielded a testing accuracy of 87.00%. The dynamics of the model's convergence process during training can be observed directly in the Figure 7, which displays the reduction of loss values alongside the improvement in accuracy over the course of 40 epochs. The loss graph (left) indicates that the training loss decreased very rapidly and consistently, starting from a value of approximately 1.30 in the early epochs and reaching a very low value in the range of 0.13-0.15 by the 40th epoch. Validation loss also decreases fairly rapidly over the first 5 epochs, from around 0.97

to 0.50, but then experiences a slowdown in convergence and stabilizes around 0.40. There is a widening gap between training loss and validation loss as the number of epochs increases, indicated by the shaded orange area on the graph.

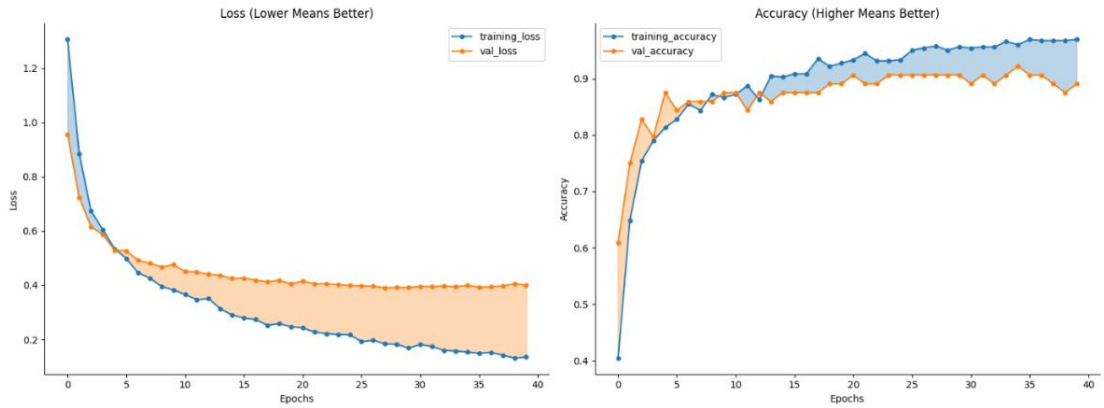

Figure 7. MobileNetV2 training curves – accuracy and loss per epoch.

In the accuracy graph (right), training accuracy shows a very significant increase from around 40% in the first epoch to a peak value of 96.95% at the end of training (epoch 40). Meanwhile, validation accuracy shows a slower upward trend with greater fluctuations, starting from around 61% and reaching a best value of 92.19% at a certain epoch during training, with a final value at the 40th epoch of 89.06%. This pattern indicates mild overfitting, where the MobileNetV2 model began to over-adapt to the training data, although the validation accuracy remained at a satisfactory level. This gap between training and validation accuracy is typical of fine-tuning a high-capacity pretrained network on a comparatively small training set (524 images), where the model can begin to memorize training-specific patterns once the more easily learned features have been captured. Common strategies for narrowing this gap in future iterations of this work include adding dropout layers before the final classification layer, applying L2 weight decay to the trainable parameters, and using early stopping based on validation loss rather than training for a fixed number of epochs, all of which are established regularization techniques for mitigating overfitting in deep convolutional networks trained on limited data [33].

In addition to the graphs, to evaluate the performance of the MobileNetV2 model in greater detail for each disease class, Figure 8 presents the confusion matrix obtained from the results of testing on the test set, which consists of 66 images (17 images per class). Based on Figure 8, the MobileNetV2 model successfully classified 59 out of 66 images correctly (86.76% ≈ 87.00%). The Red Rust class showed the best per-class performance, with 16 out of 17 images classified correctly (94.12%), followed by the Algal Leaf Spot class with 15 out of 17 images classified correctly (88.24%). The Healthy Leaf and Insects Eaten classes each showed 14 out of 17 images classified correctly (82.35%). The misclassification analysis shows that the Healthy Leaf class had the highest number of misclassifications, with one sample predicted as Algal Leaf Spot, one as Insects Eaten, and one as Red Rust. This may be due to variations in the color of healthy leaves, which sometimes resemble the early stages of disease infection. The Insects Eaten class also showed misclassification, with one sample predicted as Algal Leaf Spot and two samples predicted as Healthy Leaf, likely because minor insect damage is difficult to distinguish from healthy leaves in visual analysis.

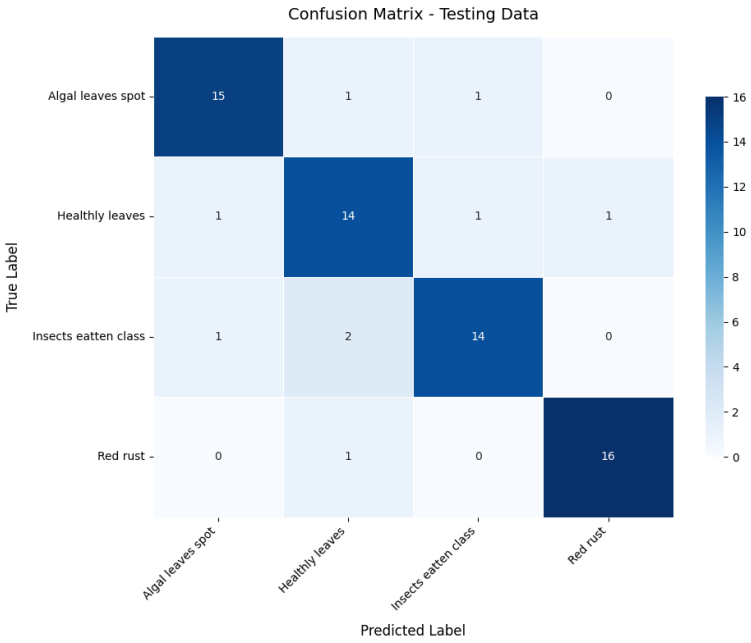

Figure 8. MobileNetV2 confusion matrix – best configuration.

Based on Figure 8, the MobileNetV2 model successfully classified 59 out of 66 images correctly (86.76% ≈ 87.00%). The Red Rust class showed the best per-class performance, with 16 out of 17 images classified correctly (94.12%), followed by the Algal Leaf Spot class with 15 out of 17 images classified correctly (88.24%). The Healthy Leaf and Insects Eaten classes each showed 14 out of 17 images classified correctly (82.35%). The misclassification analysis shows that the Healthy Leaf class had the highest number of misclassifications, with one sample predicted as Algal Leaf Spot, one as Insects Eaten, and one as Red Rust. This may be due to variations in the color of healthy leaves, which sometimes resemble the early stages of disease infection. The Insects Eaten class also showed misclassification, with one sample predicted as Algal Leaf Spot and two samples predicted as Healthy Leaf, likely because minor insect damage is difficult to distinguish from healthy leaves in visual analysis.

Table 6. Per-Class Classification Report for MobileNetV2 (Best Configuration).

Class	Precision (%)	Recall (%)	F1-Score (%)	Support
Algal Leaf Spot	88.00	88.00	88.00	17
Insects Eaten	88.00	82.00	85.00	17
Red Rust	94.00	94.00	94.00	17
Healthy Leaf	78.00	82.00	80.00	17
Average	87.00	87.00	87.00	68

3.4. Best MobileNetV3-small model performance analysis.

The best MobileNetV3-Small model was obtained in Scenario 13 with the configuration Epoch 80, Batch Size 16, Learning Rate 0.001, and Adam optimizer, achieving a test accuracy of 91.00%. Figure 9 presents the training curves for the MobileNetV3-Small model at the optimal configuration. The training loss graph (left) indicates that the training loss decreases rapidly during the first 10 epochs (from ~1.40 to approximately 0.55), then continues to decline gradually and consistently until reaching a very low value of ~0.10–0.15 at the 80th epoch. Meanwhile, the validation loss decreases at the beginning of training but then oscillates with a fairly large amplitude (±0.10) around the range of 0.50–0.65 throughout the remainder of the training process. The wider gap between the training loss and validation loss compared to

MobileNetV2 reflects the higher complexity of the MobileNetV3-Small architecture in optimizing its weights.

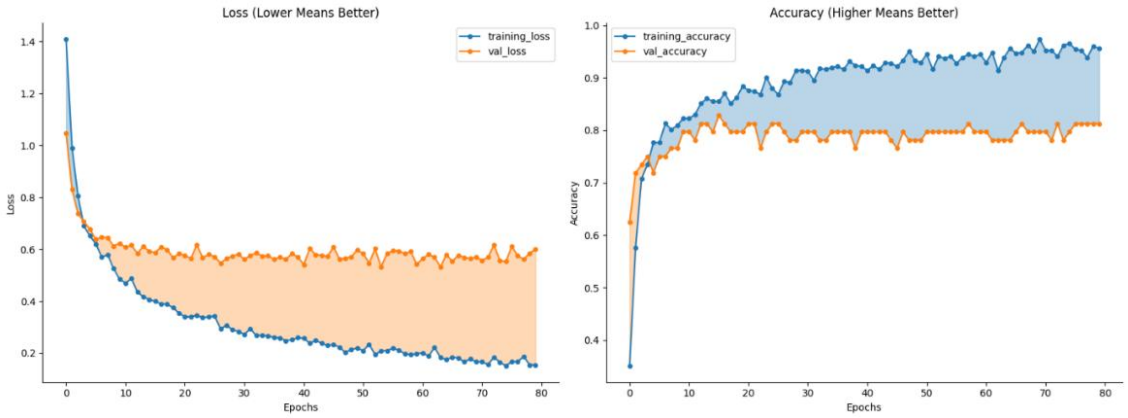

Figure 9. MobileNetV3-Small Training Curves – Accuracy and Loss per Epoch

In the accuracy plot (right), the training accuracy increased from approximately 35% in the first epoch to 95–97% by the 80th epoch. Meanwhile, the validation accuracy rose rapidly from approximately 35% to 75–80% during the first 10–15 epochs before stabilizing and fluctuating between 79% and 82% for the remainder of the training process. Although the validation accuracy remained around 79–82%, the model achieved a test accuracy of 91.00% on the independent test set. This difference is common when using relatively small datasets, where the random allocation of samples to the validation (66 images) and test (66 images) subsets may produce different data distributions, resulting in noticeable variations in performance. Previous methodological studies have reported that a single random split of a small dataset into training, validation, and test subsets can produce unstable performance estimates because the results depend on which samples are assigned to each subset rather than on deficiencies in the model itself [34]. Because both the validation and test subsets were generated using the same random splitting procedure described in Section 3.1, systematic sampling bias was considered unlikely, although it could not be completely excluded without evaluation on a larger, independently collected dataset.

To further evaluate the performance of MobileNetV3-Small for each disease class, Figure 10 presents the confusion matrix obtained from the test set. The model correctly classified 62 of the 66 test images, corresponding to an overall accuracy of 91.18% (approximately 91.00%). The Algal Leaf Spot, Healthy Leaf, and Insects Eaten classes each achieved 16 correct predictions out of 17 samples, corresponding to a recall of 94.12%, whereas the Red Rust class achieved 14 correct predictions out of 17 samples, corresponding to a recall of 82.35%. The Red Rust class exhibited the highest number of misclassifications, with two samples incorrectly classified as Algal Leaf Spot and one sample classified as Healthy Leaf. This confusion was likely caused by the visual similarity between early-stage red rust lesions and algal leaf spot symptoms, which often share reddish-brown discoloration. The Algal Leaf Spot class recorded the lowest precision (80.00%) because four samples from other classes (one Healthy Leaf, one Insects Eaten, and two Red Rust) were incorrectly predicted as Algal Leaf Spot. In contrast, the Insects Eaten and Red Rust classes achieved a precision of 100%, indicating that every sample predicted as belonging to these classes was correctly classified.

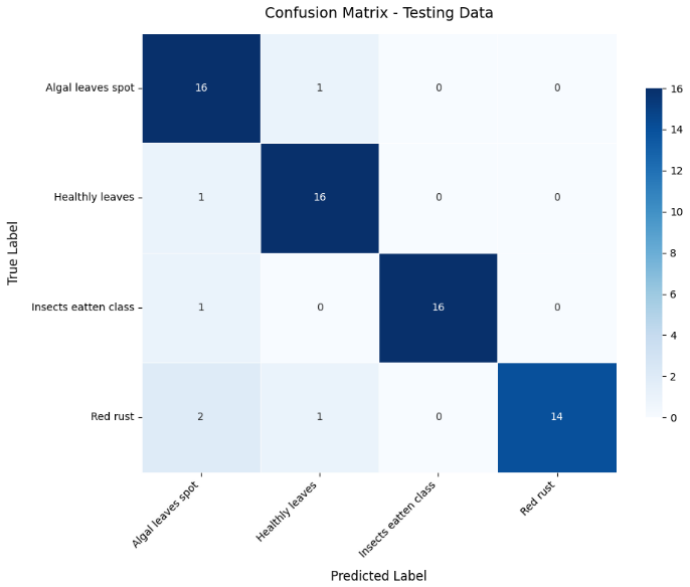

Figure 10. MobileNetV3-small confusion matrix – best configuration.

Based on Table 7, the Insects Eaten and Healthy Leaf classes achieved the highest F1-scores, both reaching 91%, followed by the Red Rust class with 90% and the Algal Leaf Spot class with 86%. The Insects Eaten class achieved a precision of 100%, indicating that the model produced no false-positive predictions for this class and successfully learned the distinctive features associated with insect damage. Similarly, the Red Rust class achieved a precision of 100% but a lower recall of 82%, suggesting that some Red Rust samples with less distinctive visual characteristics were misclassified. Overall, MobileNetV3-Small achieved higher F1-scores across all classes than MobileNetV2, with the greatest improvements observed for the Insects Eaten (+6%) and Healthy Leaf (+11%) classes. These results indicated that MobileNetV3-Small provided a better balance between precision and recall, reflecting stronger generalization capability and making it more suitable for practical plant disease detection, where reliable positive identification is essential.

Table 7. Per-Class classification report for MobileNetV3-Small (best configuration)

Class	Precision (%)	Recall (%)	F1-Score (%)	Support
Algal Leaf Spot	80.00	94.00	86.00	17
Insects Eaten	100	94.00	91.00	17
Red Rust	100	82.00	90.00	17
Healthy Leaf	89.00	94.00	91.00	17
Average	92.00	91.00	91.00	68

3.5. Model efficiency.

In addition to classification performance, this study evaluated the computational efficiency of both architectures to assess their suitability for deployment on resource-constrained devices. The efficiency metrics included the number of model parameters, the number of floating-point operations (FLOPs), and the model size. A comparison of the computational efficiency of MobileNetV2 and MobileNetV3-Small is presented in Table 8. MobileNetV2 demonstrated more stable generalization performance than MobileNetV3-Small. This was reflected by its higher validation accuracy of 92.19%, compared with 82.81% for MobileNetV3-Small. Although MobileNetV3-Small achieved a slightly higher training accuracy (97.33%), the larger gap between its training and validation accuracies suggested a greater tendency toward overfitting. This behavior was likely associated with the relatively small number of trainable

parameters (2,308), which limited the capacity of the fine-tuned classification layers and reduced the model's ability to generalize beyond the training data. It should be noted that the accuracy values reported in Table 8 were obtained from a single training and evaluation run for each configuration rather than from repeated runs with different random seeds. Furthermore, both the validation and test subsets contained only 66 images. With such a limited evaluation sample, a single random data split can produce noticeable variations in performance estimates due to sampling variability alone [34]. Therefore, the comparison between MobileNetV2 and MobileNetV3-Small should be interpreted as indicative rather than statistically conclusive. Future studies should incorporate repeated experiments, k-fold cross-validation, or confidence interval estimation to determine whether the observed differences in validation stability and classification performance are statistically significant.

Table 8. Comparison of the efficiency of the MobileNetV2 and MobileNetV3-small models.

Metric	MobileNetV2	MobileNetV3-Small
Total Class	4	4
Best Training Accuracy	0.9695 (96.95%)	0.9733 (97.33%)
Best Validation Accuracy	0.9219 (92.19%)	0.8281 (82.81%)
Total Parameters	711.348	941.428
Trainable Parameters	5.124	2.308
Model Size (Estimation)	2.71 MB	3.59 MB
FLOPs	199.233.256	116.014.816
MACs	99.616.628	58.007.408
Average Latency	18.23 ms	19.88 ms
Frame Per Second (FPS)	54.86	50.30

Architecturally, there is an intriguing paradox among model size, theoretical computational load, and real-time inference speed. Despite bearing the name "Small," MobileNetV3-Small physically has a larger estimated size (3.59 MB) than MobileNetV2 (2.71 MB). This occurs because its architectural focus was to reduce mathematical operations (FLOPs) for power efficiency, rather than merely shrinking the file size. MobileNetV3-Small indeed recorded approximately 41% lower FLOPs; however, its processing time is paradoxically slower (19.88 ms) compared to MobileNetV2 (18.23 ms). This anomaly arises because MobileNetV3 adopts the *Hard-Swish* activation function and *Squeeze-and-Excitation* (SE) modules, which trigger a high memory access load (*memory-bound bottleneck*). This heavily contrasts with the *ReLU6* activation function in MobileNetV2, which is significantly lighter and optimally executed by hardware. This observation is consistent with the original SE-Networks study, which reports that although squeeze-and-excitation adds only a small increase in FLOPs and parameters, the additional global-pooling and fully-connected operations introduce measurable extra latency in practice that is not reflected by FLOPs counts alone, particularly on hardware where these operations are not as well optimized as standard convolutions [28]. In other words, FLOPs primarily measure arithmetic operation count and do not capture memory-access overhead, so a model with fewer FLOPs can still be slower in wall-clock inference time if its operations are less hardware-friendly, as appears to be the case for MobileNetV3-Small's SE blocks relative to MobileNetV2's simpler ReLU6-based design in this study's deployment environment.

Overall, this efficiency analysis shows that MobileNetV2 is the more rational choice specifically when computational efficiency and inference speed are the primary deployment criteria. The model offers more stable validation performance in this scenario and is also capable of processing images at a higher speed, reaching 54.86 Frames Per Second (FPS), an advantage achieved because MobileNetV2 avoids the memory-access bottleneck that,

ironically, burdens its successor's SE-based architecture. This finding should be read alongside the classification-accuracy results discussed in Section 3.3, where MobileNetV3-Small achieved a higher test accuracy than MobileNetV2 (91.00% vs. 87.00%); the two models therefore represent a genuine accuracy-versus-efficiency trade-off rather than one model being unconditionally superior, and the appropriate choice between them depends on whether a deployment scenario prioritizes raw classification accuracy or inference speed and computational economy.

3.6. Discussion.

The results demonstrated that MobileNetV3-Small outperformed MobileNetV2 in classifying guava leaf diseases, achieving a higher test accuracy (91.00% versus 87.00%). This improvement was attributed to the Squeeze-and-Excitation (SE) module and the h-swish activation function, which enhanced the model's ability to emphasize informative features and identify subtle visual patterns. Although the achieved accuracy was lower than the >99% reported in some previous studies, it was consistent with studies that used similarly limited datasets (656 images). Specifically, previous research on guava leaf disease classification using a lightweight MobileNet-based model reported an accuracy of approximately 92%, which was comparable to the result obtained in this study [35]. In contrast, studies reporting accuracies exceeding 99% for plant disease classification generally employed substantially larger and more balanced datasets, such as PlantVillage, together with more extensive data augmentation strategies or additional architectural enhancements beyond a single lightweight backbone [12]. Therefore, the relatively small dataset used in this study likely limited the diversity of disease characteristics available during training, contributing to the difference between the achieved accuracy and the higher accuracies reported on larger benchmark datasets.

A class-level analysis showed that MobileNetV3-Small achieved the greatest improvements in the Healthy Leaf (+11%) and Insects Eaten (+6%) classes, indicating superior capability in distinguishing subtle visual differences. However, the model achieved a lower recall for the Red Rust class because early-stage red rust symptoms shared similar visual characteristics with Algal Leaf Spot. In terms of computational efficiency, although MobileNetV3-Small theoretically reduced the number of floating-point operations (FLOPs) by 41%, its actual inference speed was lower than that of MobileNetV2 (50.30 versus 54.86 FPS). This reduction in runtime performance was likely caused by the additional memory access overhead introduced by the SE modules.

Despite several limitations, including the relatively small dataset and the use of images collected from a single regional source, this study established a reproducible baseline for guava leaf disease classification using lightweight deep learning models. The limited dataset size, restricted geographic diversity, and inclusion of only four disease classes may have constrained the models' ability to generalize to images captured under different environmental conditions, growth stages, or disease manifestations. Furthermore, the reported performance has not yet been validated using an independent external dataset, which remains an important step before practical field deployment.

Future research should focus on expanding the dataset with more diverse guava leaf images collected from multiple regions in Indonesia, comparing MobileNetV3-Small with other lightweight architectures such as EfficientNet-Lite, and incorporating Explainable Artificial Intelligence (XAI) techniques such as Grad-CAM to improve model interpretability.

Integrating the trained model into a smartphone- or IoT-based diagnostic system would also enable rapid field diagnosis while providing visual explanations of the image regions used for prediction. Such visual evidence could help farmers and agricultural extension workers verify that the model based its decisions on disease symptoms rather than background artifacts, thereby improving confidence in AI-assisted plant disease diagnosis systems [36].

4. Conclusions

This study proposed MobileNetV3-Small for guava leaf disease classification and systematically compared its performance with MobileNetV2 using a transfer learning approach on a balanced dataset of 656 guava leaf images representing four classes: Algal Leaf Spot, Insects Eaten, Red Rust, and Healthy Leaf. The experimental results showed that MobileNetV3-Small achieved the highest test accuracy of 91.00% using 80 epochs, a batch size of 16, a learning rate of 0.001, and the Adam optimizer, outperforming MobileNetV2 by 4%, which achieved a best accuracy of 87.00%. The incorporation of Squeeze-and-Excitation (SE) modules and the h-swish activation function enabled MobileNetV3-Small to learn more discriminative features, resulting in notable improvements in the Healthy Leaf (+11% F1-score) and Insects Eaten (+6% F1-score) classes. From a computational perspective, MobileNetV3-Small reduced the number of floating-point operations (FLOPs) by approximately 41% compared with MobileNetV2. However, MobileNetV2 retained a slight advantage in inference speed (54.86 FPS versus 50.30 FPS) and model size (2.71 MB versus 3.59 MB) because the SE modules introduced additional memory-access overhead during inference. These findings indicate that MobileNetV3-Small is more suitable for applications where classification accuracy is the primary objective, whereas MobileNetV2 remains an attractive alternative for latency-sensitive deployment on resource-constrained devices. Although the proposed model achieved promising performance, the study was limited by the relatively small dataset and the absence of external validation using independently collected field images. Future work should therefore expand the dataset with more diverse guava leaf images collected from different regions of Indonesia, evaluate additional lightweight architectures, incorporate explainable AI techniques such as Grad-CAM, and validate the model under real-world field conditions. Integrating the trained model into a smartphone- or IoT-based diagnostic application could provide farmers with rapid and low-cost preliminary disease identification while supporting agricultural extension officers in prioritizing field inspections and improving disease management. These developments would enhance the practical applicability of artificial intelligence for sustainable and precision agriculture.

Acknowledgments

The author expresses sincere gratitude to Mrs. Ery Hartati for her guidance and valuable feedback throughout this research.

Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have influenced the work reported in this article.

References

[1] Anam, F.S.; Muttaqin, M.R.; Ramadhan, Y.R. (2023). Klasifikasi penyakit pada daun dan buah jambu menggunakan convolutional neural network. *JOINTECS (Journal of Information Technology and Computer Science)*, *8*(3), 115. http://doi.org/10.31328/jointecs.v8i3.4823.

[2] Badan Pusat Statistik. Produksi Jambu Biji Indonesia. (accessed on 27 June 2026) Available online: https://www.bps.go.id/id/statistics-table/3/U0dKc1owczVSalJ5VFdOMWVETnlVRVJ6YlRJMFp6MDkjMw==/produksi-tanaman-buah-buahan-dan-sayuran-tahunan-menurut-provinsi-dan-jenis-tanaman--2024.html?year=2024.

[3] Bakara, F.K.R.D.J. (2020). Pendampingan petani dalam pengendalian hama dan penyakit jambu biji (*Psidium guajava* L.) di Desa Cibening, Pamijahan. *Jurnal Pusat Inovasi Masyarakat*, *2*(1), 131–143.

[4] Yamin, M.; Sulastri, M.A.; Damayanthy, D. (2025). Enhancing agricultural productivity and food security through Climate Smart Agriculture (CSA) adoption: The interplay of social, economic and environmental in tidal swamp farming. *AGRIS On-line Papers in Economics and Informatics*, *17*(3), 117–131. http://doi.org/10.7160/aol.2025.170310.

[5] Orjiakor, E.C. (2025). Securing food systems: Pathways of COVID-19 impact on food security for smallholder farmers in Nigeria. *Journal of Hunger & Environmental Nutrition*, 1–12. http://doi.org/10.1080/19320248.2025.2564145.

[6] Kurniawan, D.; Apriandari, W.; Pambudi, A. (2023). Sistem pakar diagnosa hama penyakit tanaman jambu kristal penerapan metode certainty factor. *Bit (Fakultas Teknologi Informasi Universitas Budi Luhur)*, *20*(2), 168–177. http://doi.org/10.36080/bit.v20i2.2462.

[7] Lv, Q.; Zhang, S.; Wang, Y. (2022). Deep learning model of image classification using machine learning. *Advances in Multimedia*, *2022*. http://doi.org/10.1155/2022/3351256.

[8] Syahputra, F. (2023). Analisis arsitektur deep learning MobileNet dalam mengklasifikasi hama daun jambu madu. Bachelor's Thesis, Universitas Medan Area, Medan, Indonesia.

[9] Maximilliano, W.; Rachmat, N. (2025). Comparative analysis of MobileNetV3-Large and small for corn leaf disease classification. *Brilliance: Research of Artificial Intelligence*, *5*(1), 325–332. http://doi.org/10.47709/brilliance.v5i1.6259.

[10] Yin, X.; Li, W.; Li, Z.; Yi, L. (2022). Recognition of grape leaf diseases using MobileNetV3 and deep transfer learning. *International Journal of Agricultural and Biological Engineering*, *15*(3), 184–194. http://doi.org/10.25165/j.ijabe.20221503.7062.

[11] Zhang, J.; Yang, X.; Fu, X.; Wang, B.; Li, H. (2025). LDL-MobileNetV3S: An enhanced lightweight MobileNetV3-small model for potato leaf disease diagnosis through multi-module fusion. *Frontiers in Plant Science*, *16*, 1–19. http://doi.org/10.3389/fpls.2025.1656731.

[12] Duhan, S.; Gulia, P.; Gill, N.S.; Oliveira, T.A.; Singh, P.K. (2026). Evaluation of lightweight and efficient deep learning models for plant disease classification. *Discover Internet of Things*, *6*(1). http://doi.org/10.1007/s43926-026-00310-0.

[13] Xu, Y.; Li, D.; Li, C.; Yuan, Z.; Dai, Z. (2025). LiSA-MobileNetV2: An extremely lightweight deep learning model with Swish activation and attention mechanism for accurate rice disease classification. *Frontiers in Plant Science*, *16*, 1–14. http://doi.org/10.3389/fpls.2025.1619365.

[14] Nur Rifa'i, W.R.; Erwanto, D.; Yanuartanti, I. (2025). Evaluasi kinerja CNN dengan optimizer RMSprop, Adam dan SGD dalam klasifikasi penyakit daun anggur. *JE-Unisla*, *10*(1), 91–104. http://doi.org/10.30736/je-unisla.v10i1.1409.

[15] Mathew, A.; Amudha, P.; Sivakumari, S. (2021). Deep learning techniques: An overview. In Advanced Machine Learning Technologies and Applications, pp. 599–608; Springer Singapore: Singapore.

[16] Nurhakiki, J.; Yahfizham, Y. (2024). Studi kepustakaan: Pengenalan 4 algoritma pada pembelajaran deep learning beserta implikasinya. *Pendekar: Jurnal Pendidikan Berkarakter*, *2*(1), 270–281. http://doi.org/10.51903/pendekar.v2i1.598.

[17] Ramadhan, M.F.; Fachrie, M. (2024). Implementasi algoritma Long Short-Term Memory (LSTM) pada sistem prediksi hasil panen sawit. *Jurnal Informatika Teknologi dan Sains*, *6*(4), 937–944. http://doi.org/10.51401/jinteks.v6i4.4876.

[18] Alzubaidi, L.; et al. (2021). Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, *8*(1), 53. http://doi.org/10.1186/s40537-021-00444-8.

[19] Salka, T.D.; Hanafi, M.B.; Syed Ahmad Abdul Rahman, S.M.; Mohamed Zulperi, D.B.; Omar, Z. (2025). Plant leaf disease detection and classification using convolutional neural networks model: A review. *Artificial Intelligence Review*, *58*, 322. http://doi.org/10.1007/s10462-025-11234-6.

[20] Zhao, X.; Wang, L.; Zhang, Y.; Han, X.; Deveci, M.; Parmar, M. (2024). A review of convolutional neural networks in computer vision. *Artificial Intelligence Review*, *57*(4), 99. http://doi.org/10.1007/s10462-024-10721-6.

[21] Dhaka, V.S.; et al. (2021). A survey of deep convolutional neural networks applied for prediction of plant leaf diseases. *Sensors*, *21*(14). http://doi.org/10.3390/s21144749.

[22] Georgieva, P. (2019). Deep learning in brain computer interfaces. In *ACM International Conference Proceeding Series*. http://doi.org/10.1145/3351556.3351594.

[23] Howard, A.; et al. (2019). Searching for MobileNetV3. *arXiv preprint*. http://doi.org/10.48550/arXiv.1905.02244.

[24] Gao, Y.; Jiang, Y.; Peng, Y.; Yuan, F.; Zhang, X.; Wang, J. (2025). Medical Image Segmentation: A Comprehensive Review of Deep Learning-Based Methods. *Tomography*, *11*, 52. https://doi.org/10.3390/tomography11050052.

[25] Ekmekyapar, T.; Taşcı, B. (2023). Exemplar MobileNetV2-Based Artificial Intelligence for Robust and Accurate Diagnosis of Multiple Sclerosis. *Diagnostics*, *13*, 3030. https://doi.org/10.3390/diagnostics13193030.

[26] Khan, A.T.; Jensen, S.M.; Khan, A.R.; Li, S. (2023). Plant disease detection model for edge computing devices. *Frontiers in Plant Science*, *14*, 1–10. http://doi.org/10.3389/fpls.2023.1308528.

[27] Dhanalaxmi, B.; Kumar, B.N.; Raju, Y.; Channapragada, R.S.R. (2025). MobileNetV3: An efficient deep learning-based feature selection and classification technique for cardiovascular disease. *Journal of Engineering and Applied Science*, *72*(1), 1–33. http://doi.org/10.1186/s44147-025-00654-4.

[28] Hu, J.; Shen, L.; Sun, G. (2018). Squeeze and excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7132–7141.

[29] Hosna, A.; Merry, E.; Gyalmo, J.; Alom, Z.; Aung, Z.; Azim, M.A. (2022). Transfer learning: A friendly introduction. *Journal of Big Data*, *9*(1). http://doi.org/10.1186/s40537-022-00652-w.

[30] Altınel, D. (2025). Development of deep learning optimizers: Approaches, concepts, and update rules. *arXiv preprint*. http://doi.org/10.48550/arXiv.2509.18396.

[31] Dogo, E.M.; Afolabi, O.J.; Twala, B. (2022). On the relative impact of optimizers on convolutional neural networks with varying depth and width for image classification. *Applied Sciences*, *12*(23). http://doi.org/10.3390/app122311976.

[32] Yamasari, Y.; Sadewa, B.A.; Qoiriah, A.; Yohannes, E.; Putra, R.E.; Ahmad, T. (2025). Hyperparameter analysis of Adam-optimized deep transfer learning for Indonesian banknote-denomination recognition. *Journal of Hunan University Natural Sciences*, *52*(5). http://doi.org/10.55463/issn.1674-2974.52.5.19.

[33] Tian, Y.; Zhang, Y. (2022). A comprehensive survey on regularization strategies in machine learning. *Information Fusion*, *80*, 146–166. http://doi.org/10.1016/j.inffus.2021.11.005.

[34] An, C.; Park, Y.W.; Ahn, S.S.; Han, K.; Kim, H.; Lee, S.K. (2021). Radiomics machine learning study with a small sample size: Single random training-test set split may lead to unreliable results. *PLOS ONE*, *16*(8), 1–13. http://doi.org/10.1371/journal.pone.0256152.

[35] Mustak Un Nobi, M.; Rifat, M.; Mridha, M.F.; Alfarhood, S.; Safran, M.; Che, D. (2023). GLD-Det: Guava leaf disease detection in real-time using lightweight deep learning approach based on MobileNet. *Agronomy*, *13*(9), 1–21. http://doi.org/10.3390/agronomy13092240.

[36] Pal, C.; Karmakar, S.; Mukherjee, I.; Chakrabarti, P.P. (2025). A lightweight and explainable CNN model for empowering plant disease diagnosis. *Scientific Reports*, *15*(1), 1–14. http://doi.org/10.1038/s41598-025-94083-1.