JURNAL INFOTEL
VOL. 18, NO. 3, AUGUST 2026, PP. 371–387. DOI:10.20895/INFOTEL.V18I3.11387

RESEARCH ARTICLE

Performance Evaluation of Lightweight CNN Architectures for Mobile Maize Leaf Image Classification

Sirli Fahriah[1,*], Catur Edi Widodo[2], and Prayitno[3]

[1,2]Doctoral Program of Information System, Diponegoro University, Semarang 50241, Indonesia
[3]Department of Electrical Engineering, Politeknik Negeri Semarang, Semarang 50275, Indonesia

*Corresponding email: sirlifahriah@student.undip.ac.id

Received: June 15, 2026; Revised: August 21, 2026; Accepted: September 12, 2026.

Abstract: This study evaluates three lightweight Convolutional Neural Network (CNN) architectures for maize leaf disease classification and assesses the feasibility of Android deployment. The dataset contained 2,760 images across six classes: Leaf Spot, Downy Mildew, Blight, Rust, Healthy, and Non-leaf. It included 2,237 primary images collected from agricultural field observations and 523 secondary images from public datasets and web-based collections, including 453 Non-leaf images. MobileNetV2, NasNetMobile, and EfficientNetB0 were evaluated using a consistent protocol. In hold-out testing, MobileNetV2 achieved the highest accuracy (97.10%) and macro F1-score (0.970), compared with 91.30% and 0.908 for NasNetMobile and 95.65% and 0.953 for EfficientNetB0. Five-fold cross-validation yielded a mean MobileNetV2 validation accuracy of 95.51% ± 0.92%. The selected model was converted to TensorFlow Lite and deployed in an Android application; testing on 30 previously unseen images produced 29 correct predictions (96.67%). These results support MobileNetV2 for this application based on the evaluated predictive performance and functional deployment. However, comprehensive mobile efficiency benchmarking was not possible because CPU and NNAPI latency, RAM consumption, and model size were not systematically measured. Future work should assess these on-device metrics to better characterize deployment efficiency.

Keywords: lightweight CNN; maize leaf disease classification; MobileNetV2; Android deployment; TensorFlow Lite; on-device inference.

© Jurnal Infotel ISSN: 2085-3688; e-ISSN: 2460-0997

1 Introduction

The increasing use of artificial intelligence (AI) in mobile and edge applications has encouraged the deployment of image classification models directly on resource-constrained devices. On-device inference can reduce dependence on network connectivity and support responsive image analysis, but mobile deployment also introduces constraints related to computational complexity, memory consumption, and inference efficiency [1–3]. These constraints make the selection of the CNN architecture important when an image classification model is intended for smartphone-based applications.

Therefore, lightweight CNN architectures have been developed to reduce computational requirements while maintaining competitive predictive performance. MobileNetV2 employs inverted residual blocks and linear bottlenecks to achieve an efficient architecture for resource-constrained applications [4]. EfficientNet uses compound scaling to balance network depth, width, and input resolution [5], while NASNet is based on neural architecture search and provides architectures designed to transfer learned architectural patterns across image recognition tasks [6]. These architectural differences make MobileNetV2, NasNetMobile, and EfficientNetB0 relevant candidates for comparative evaluation in mobile-oriented image classification.

Maize leaf disease classification provides a relevant agricultural computer vision task for evaluating these architectures because disease symptoms can exhibit visually similar patterns and practical image acquisition may introduce variability beyond controlled datasets. Previous studies have demonstrated the effectiveness of deep learning and transfer learning for maize disease classification [7–10], while recent research has also investigated lightweight CNN architectures specifically for maize disease recognition [11–18]. These studies demonstrate the potential of deep learning for automated maize disease recognition while also highlighting the importance of computational efficiency for practical applications.

Mobile and edge deployment has also been investigated in recent plant disease research. Khan et al. developed a mobile-based maize disease detection and classification system using deep learning [9]. Other studies have integrated deep learning models with mobile applications and TensorFlow Lite for plant disease classification [1, 19, 20]. In maize disease recognition specifically, recent work has investigated MobileNetV2 with TensorFlow Lite for mobile deployment [21], while TinyML and post-training quantization approaches have explored computational and memory constraints for resource-limited agricultural devices [18, 22]. These studies indicate that mobile deployment and model optimization are established research directions rather than unexplored problems.

However, performance reported across these studies is not always directly comparable because datasets, disease categories, preprocessing procedures, training configurations, hardware platforms, and evaluation protocols differ. Some studies focus on developing a new lightweight architecture [14, 15, 17], whereas others focus on deploying a selected model to a mobile application [1, 19, 21]. Consequently, results from independent studies cannot be directly interpreted as evidence that one architecture is universally superior. A controlled comparison of multiple lightweight pretrained CNN architectures under a consistent dataset, preprocessing procedure, training configuration, and evaluation protocol can therefore provide more meaningful evidence regarding their relative predictive performance.

In addition to disease categories, practical mobile image acquisition may produce irrelevant images that do not contain a maize leaf. A classifier trained exclusively on disease categories may consequently assign such an image to one of the target classes. In this study, a Non-leaf class is included to represent irrelevant visual input outside the target maize-leaf domain. This additional class provides a practical robustness dimension by evaluating whether the models can distinguish relevant maize leaf images from non-target inputs.

Although classification accuracy remains an important indicator of model effectiveness, it does not fully characterize the suitability of a CNN architecture for mobile deployment. A model with higher predictive performance may require greater computational resources, whereas a more compact architecture may provide more favorable deployment characteristics. Therefore, evaluating lightweight CNNs should consider predictive performance together with the available evidence on model complexity and deployment feasibility [2, 3, 13, 18, 22]

2 Research Method

2.1 Related Work

Deep learning-based image classification has been widely applied to the recognition of maize leaf disease because convolutional neural networks (CNNs) can automatically learn discriminative visual features from leaf images. Previous studies have demonstrated the effectiveness of CNNs and transfer learning for maize disease classification using pretrained CNN models and other deep learning approaches [7, 8, 10]. More recent studies have further explored transfer learning and lightweight architectures to improve classification performance while reducing computational requirements [9–15].

Lightweight CNN architectures are particularly relevant for applications intended for resource-constrained devices. MobileNetV2 uses inverted residual blocks and linear bottlenecks to reduce computational complexity while maintaining an effective feature representation [4]. EfficientNet introduces compound scaling of network depth, width, and input resolution to achieve a favorable balance between accuracy and computational efficiency [5]. NASNet is based on the search for neural architectures and provides architectures designed through automatically learned network structures [6]. Other studies have investigated lightweight architectures specifically for the recognition of maize disease, including improved MobileNet, ShuffleNetV2, and lightweight residual architectures [13–15]. These studies indicate that lightweight CNN architectures are promising candidates for practical agricultural image classification.

Recent research has also increasingly considered mobile and edge deployment. Khan et al. developed a mobile maize disease detection system in which a deep learning model was integrated into a mobile application [9]. Other studies have investigated mobile applications for the diagnosis of plant disease using deep learning and TensorFlow Lite [1, 19, 20]. Lightweight and TinyML-based approaches have also addressed deployment constraints such as computational complexity, memory requirements, and inference performance [2, 3, 18, 21, 22]. These studies demonstrate that mobile and edge deployment is an already established research direction in agricultural disease recognition.

Nevertheless, direct comparison of lightweight pretrained CNN architectures remains challenging because previous studies use different datasets, disease categories, preprocessing strategies, training configurations, evaluation protocols, and deployment environ-

ments. Some studies focus on proposing a new lightweight architecture [14, 15, 17], while others focus on deploying a particular model to a mobile application [1, 19, 21]. Consequently, performance differences reported across independent studies cannot be directly interpreted as evidence that one architecture is universally superior. A controlled comparison using the same dataset, preprocessing procedure, training configuration, and evaluation protocol can therefore provide more meaningful evidence regarding the relative performance of candidate lightweight CNN architectures.

Another practical consideration is handling irrelevant image input. Smartphone-based image acquisition may produce images that do not contain a relevant maize leaf. A classifier trained only on disease categories may consequently assign an irrelevant image to one of the disease classes. The inclusion of a Non-leaf class provides an additional robustness dimension by requiring the model to distinguish relevant maize leaf images from irrelevant input. This consideration is particularly important for mobile applications, where image acquisition conditions are less controlled than laboratory datasets.

Broader plant-disease studies have also highlighted the importance of dataset diversity and transfer learning for real-world disease recognition. The PlantDoc dataset was introduced to address the challenges associated with visual plant disease detection in less controlled environments [23]. Survey and comparative studies have also summarized the application of deep convolutional networks and transfer learning to the recognition of plant disease [24–26]. These findings reinforce the need to consider both dataset characteristics and model generalization when evaluating image-based plant disease classifiers.

Recent work has also explored compressed MobileNetV2 representations specifically for the recognition of maize leaf disease, further demonstrating the relevance of lightweight feature extraction for resource-constrained classification [27].

Based on these considerations, the contribution of this study is not the introduction of mobile deployment itself. Instead, this study provides a controlled comparison of three lightweight pretrained CNN architectures MobileNetV2, NasNetMobile, and EfficientNetB0—using the same maize image classification task and training configuration, followed by a functional Android deployment of the selected architecture. The study focuses on comparative predictive performance, available cross-validation stability evidence, robustness to the Non-leaf class, and the practical feasibility of functional Android deployment under a consistent experimental setting.

2.2 Research Framework

This study employs a deployment-oriented experimental framework to evaluate lightweight CNN architectures under a consistent image-classification and Android deployment setting. The framework consists of five sequential stages: dataset preparation, model training, predictive performance evaluation, TensorFlow Lite conversion, and functional Android deployment evaluation. First, the dataset is prepared through image verification, preprocessing, dataset partitioning, and training-only augmentation. Second, three pretrained lightweight CNN architectures MobileNetV2, NasNetMobile, and EfficientNetB0 are trained using the same classification task and training protocol. Third, predictive performance is evaluated using accuracy, precision, recall, F1-score, and available cross-validation analysis. Fourth, the trained models are converted to TensorFlow Lite format without additional quantization. Finally, the selected model is integrated into an Android application and evaluated through functional mobile testing.

The framework separates predictive evaluation from functional deployment evaluation. Classification performance is assessed using the predefined hold-out protocol, while five-fold cross-validation is used as complementary evidence of performance stability. The evaluation of Android focuses on the successful execution of the classification workflow and the results of mobile classification rather than a complete quantitative benchmark for hardware-efficiency.

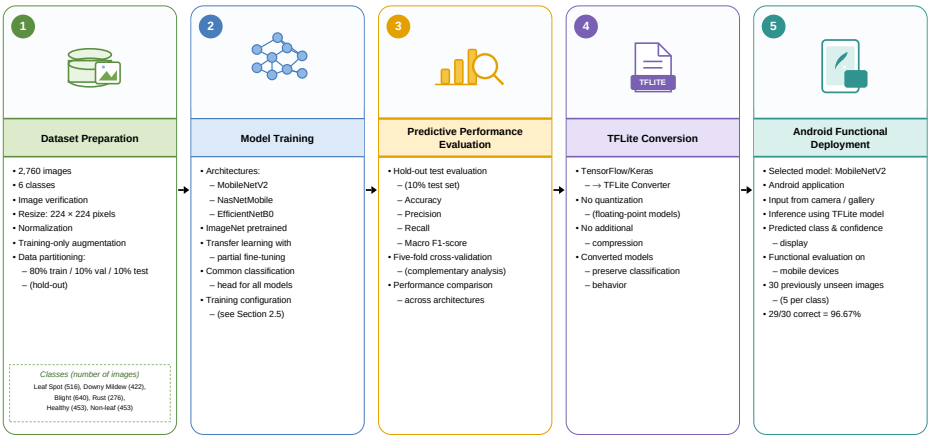

Figure 1: Research framework for evaluating lightweight CNN architectures on Android deployment.

2.3 Dataset and Preprocessing

The dataset used in this study comprises 2,760 images distributed across six classes: Leaf Spot, Downy Mildew, Blight, Rust, Healthy, and Non-leaf. The dataset was constructed from primary and secondary image sources. A total of 2,237 images were collected as primary data from maize fields in Kendal Regency, Indonesia, using smartphone cameras during field observations. The remaining 523 images were obtained from secondary sources, including publicly available datasets and web-based image collections. Disease categories were assigned according to observed visual characteristics and supported by relevant agricultural references and consultation with an agricultural expert.

The data set contains 484 Leaf Spot images, 438 Downy Mildew images, 435 Blight images, 495 ruth images, 455 Healthy images, and 453 Non-leaf images. The Non-leaf class was obtained entirely from secondary data and consists of 453 images that do not contain maize leaves. This class was intentionally included to represent irrelevant inputs that may occur during practical smartphone image acquisition and to evaluate whether the models can distinguish target maize-leaf images from non-target inputs.

Before model training, all images were screened for file validity and visual suitability. The images were resized to 224 $\times$ 224 pixels and represented using three RGB channels. Pixel normalization was applied according to the preprocessing requirements of the pre-processed architectures evaluated. Data augmentation was applied only to the training

data, whereas the validation and testing images were retained without augmentation. The augmentation procedures included image flipping, rotation, brightness adjustment, contrast variation, cropping, and resizing. The available experimental documentation confirms that the images were standardized before training and that augmentation was restricted to the training data.

Table 1: Data distribution

Class	Training	Validation	Testing	Total
Healthy	366	44	45	455
Leaf Spot	388	48	48	484
Downy Mildew	339	58	41	438
Blight	355	40	40	435
Rust	391	48	56	495
Non-leaf	369	38	46	453
Total	2,208	276	276	2,760

The dataset was partitioned using a hold-out strategy into 80% training, 10% validation and 10% testing, corresponding to 2,208, 276, and 276 images, respectively. The testing subset was reserved for the final hold-out evaluation, while the validation subset was used during model development. The relationship between the hold-out partition and the five-fold cross-validation procedure is described separately in subsection 2.6. The documented dataset split confirms the 80/10/10 distribution shown in Table 1.

2.4 Lightweight CNN Architectures

Three pre-trained CNN architectures were selected for the benchmark: MobileNetV2, NasNetMobile, and EfficientNetB0. The three architectures were selected because they provide different approaches to achieving computationally efficient image classification and are oriented toward applications with constrained computational resources. All three architectures were evaluated using the same six-class classification task and training protocol to allow a controlled comparison of their predictive and deployment characteristics.

MobileNetV2 was selected as a representative mobile-oriented CNN architecture designed to reduce computational requirements while maintaining an effective feature representation. Its architecture employs inverted residual blocks and linear bottlenecks to reduce the computational cost of convolutional operations while preserving useful feature information [4]. In this study, MobileNetV2 represents an architecture explicitly designed for efficient inference on resource-constrained devices.

NasNetMobile represents a mobile-oriented architecture developed using neural architecture search (NAS), in which network structures are optimized through an automated architecture search process. The NAS approach was introduced to learn transferable architectures for image recognition, while NasNetMobile provides a compact variant suitable for mobile-oriented image classification [6]. It was included to examine whether an architecture derived by searching for neural architecture provides a competitive balance between classification performance and deployment feasibility.

EfficientNetB0 was selected as an efficiency-oriented CNN architecture that applies compound scaling to balance network depth, width, and input resolution [5]. This scal-

ing strategy provides an alternative approach to balancing predictive representation and computational requirements. Therefore, EfficientNetB0 provides a useful comparison with MobileNetV2 and NasNetMobile when evaluating the relationship between predictive performance and deployment requirements.

To ensure a fair comparison, the three architectures were trained using the same dataset partition, preprocessing strategy, augmentation procedure, classification task, and general training protocol. Architecture-specific differences were retained only where required by the respective pretrained network implementation. The resulting models were subsequently evaluated using the same predictive-performance and deployment-oriented evaluation procedures.

2.5 Model Training Configuration

The three selected CNN architectures MobileNetV2, NasNetMobile, and EfficientNetB0 were trained using a transfer-learning strategy with ImageNet-pre-trained weights. All input images were resized to 224 $\times$ 224 pixels with three RGB channels. A common classification head consisting of Global Average Pooling, a fully connected layer with 256 units, Dropout with a rate of 0.5, and a six-class Softmax output layer was used for all architectures to maintain a consistent comparison setting.

The models were trained for a maximum of 30 epochs using the Adam optimizer with an initial learning rate of 1×10^{-4} and categorical cross-entropy loss. The batch size was set to 32. A fixed random seed of 42 was used to improve reproducibility. Early Stopping was applied with patience of five epochs, while ReduceLROnPlateau used patience of two epochs and a reduction factor of 0.5 when validation performance reached a plateau. ModelCheckpoint was used to retain the model weights associated with the best validation performance.

The convolutional bases were not completely frozen during training. Partial fine-tuning was applied to the final layers of each pretrained architecture. The last 20 layers of MobileNetV2 were trainable, while the last 10 layers of NasNetMobile and EfficientNetB0 were trainable; the preceding layers remained frozen. This configuration was applied consistently across the three architectures to provide a controlled basis for comparison.

The preprocessing and augmentation procedures were kept consistent across the models. Training images were subjected to the specified augmentation operations, whereas validation and test images were not augmented. The resulting models were evaluated using the same predefined hold-out data and evaluation procedures.

2.6 Cross-Validation and Evaluation Protocol

Five-fold stratified cross-validation was conducted for MobileNetV2 as a complementary analysis of the stability of the classification performance across different data partitions. The complete data set consisting of 2,760 images was shuffled using a fixed random seed of 42, after which a five-fold stratified partitioning was applied to maintain a comparable class distribution across folds.

In each iteration, four folds comprising 2,208 images were used as the training fold and one fold comprising 552 images was used as the validation fold. The process was repeated for five iterations by assigning a different fold as the validation subset. Consequently, every image participated in the validation subset once during the five-fold procedure.

The five-fold cross-validation procedure was conducted separately from the predefined hold-out evaluation. The hold-out protocol consisted of 80% training, 10% validation, and 10% testing data, corresponding to 2,208, 276, and 276 images, respectively. The 276-image hold-out test subset was reserved for the final test evaluation. However, because the documented five-fold procedure was performed using the complete dataset of 2,760 images, the images belonging to the hold-out test subset were also included in the cross-validation folds. Therefore, the five-fold results are not considered an independent estimate of the final hold-out test performance and were not used as a basis for final model selection or hyperparameter tuning. They are reported only as complementary evidence of performance stability across stratified data partitions.

The hold-out test results are therefore treated as the primary evaluation results, while the five-fold results provide additional information regarding the stability of MobileNetV2 across different data partitions. Equivalent fold-wise results for NasNetMobile and EfficientNetB0 are not available in the documented experimental records used for this revision; consequently, no cross-validation comparison or superiority claim is made for these two architectures.

Table 2 presents the data allocation for each cross-validation iteration. The row numbers refer to the shuffled dataset and therefore do not represent the original ordering of image acquisition or class labels.

Table 2: Five-fold cross-validation data partitioning for MobileNetV2

Fold	Training Fold	Validation Fold	Training Proportion	Validation Proportion
Fold 1	2,208	552	80%	20%
Fold 2	2,208	552	80%	20%
Fold 3	2,208	552	80%	20%
Fold 4	2,208	552	80%	20%
Fold 5	2,208	552	80%	20%

2.7 TensorFlow Lite Conversion

The trained CNN models were converted from the TensorFlow/Keras format to the TensorFlow Lite (TFLite) format to support Android deployment. The conversion was performed using the TensorFlow Lite Converter, and the resulting TFLite models were intended for integration into the Android application for mobile inference.

The conversion used the baseline floating-point TFLite representation without additional post-training quantization or compression. Therefore, no quantization of INT8 or float16 is claimed as part of the implemented experiment. The conversion was treated as a model-format transformation for mobile deployment rather than as an optimization procedure.

The classification results reported in the hold-out evaluation were obtained from the original trained models and are not interpreted as post-conversion accuracy measurements. Similarly, quantitative measurements of the size reduction of the TFLite model and inference delay are not reported because the corresponding measurements are not available in the documented experimental records. Therefore, the Android evaluation focuses on

functional execution of the converted model, while quantitative benchmarking of mobile efficiency is identified as a limitation of the present study.

2.8 Android Deployment and Measurement

The selected MobileNetV2 model was integrated into an Android application using its converted TensorFlow Lite representation to evaluate the functional deployment on smartphone devices. The application accepts an image input from the device camera or gallery, performs the required image preprocessing, and executes CNN inference using the TFLite model. The predicted class and confidence score are then presented as the classification output.

The functional deployment was evaluated on two Android smartphones, namely the Infinix Smart 6 and Redmi Note 12, to verify that the application and inference pipeline could operate on different hardware configurations. The evaluation focused on the successful execution of the image-input, preprocessing, inference, and prediction-output workflow rather than a comprehensive quantitative hardware benchmark.

A separate mobile classification test was conducted using 30 previously unseen images, five images representing each of the six target classes. These images were used to verify that the deployed application could produce classification outputs on mobile devices. The mobile test was treated as a functional deployment assessment and was reported separately from the predefined hold-out evaluation.

Quantitative measurements of CPU latency, NNAPI latency, RAM consumption, thermal behavior, and TFLite model-size reduction were not available in documented experimental records. Therefore, these metrics are not presented as measured results in this study. The evaluation of Android is therefore interpreted as evidence of functional deployment, rather than as a complete benchmark of mobile computational efficiency.

3 Results

This section presents the experimental results based on the methodology described in section 2. The evaluation focuses on both classification performance and deployment feasibility to identify the most suitable model for real-world implementation. The results presented in this section directly correspond to the evaluation framework described in section 2, which ensures the consistency between the experimental design and the performance analysis.

3.1 Cross-Validation Results

Five-fold stratified cross-validation was conducted for MobileNetV2 to examine the stability of its classification performance across different data partitions. The complete dataset was shuffled and stratified into five folds, with 2,208 images used for training and 552 images used for validation in each iteration. Therefore, each image participated in the validation subset once in the five iterations.

The validation accuracies obtained for MobileNetV2 across the five folds were 94.75%, 94.57%, 96.38%, 94.93%, and 96.92%, respectively. The resulting mean validation accuracy was 95.51%, with a standard deviation of 0.92 percentage points. The relatively small vari-

ation across the five folds indicates stable validation performance under different stratified data partitions.

The cross-validation analysis is interpreted as an assessment of performance stability rather than as an independent estimate of the final generalization. Since equivalent fold-wise results for NasNetMobile and EfficientNetB0 are not available in the documented experimental records used for this revision, no cross-validation superiority claim is made for these two architectures.

Table 3: Five-fold cross-validation results for MobileNetV2

Fold	Training Accuracy (%)	Validation Accuracy (%)
Fold 1	95.06	94.75
Fold 2	95.74	94.57
Fold 3	93.32	96.38
Fold 4	94.78	94.93
Fold 5	95.50	96.92
Mean	94.88	95.51
Std. Dev.	0.91	0.92

The validation accuracy remained relatively consistent across the five folds, with all folds achieving accuracy above 94%. This indicates that MobileNetV2 maintained relatively stable classification performance under different validation partitions. However, the cross-validation results are interpreted as an assessment of performance stability rather than as an independent estimate of the final generalization. Because the cross-validation procedure was conducted using the complete dataset, its results are reported separately from the predefined hold-out evaluation.

3.2 Hold-Out Classification Performance

Table 4 summarizes the hold-out classification performance of the three lightweight CNN architectures. MobileNetV2 achieved the highest test accuracy of 97.10%, followed by EfficientNetB0 at 95.65% and NasNetMobile at 91.30%. MobileNetV2 also achieved the highest macro precision, macro recall, and macro F1-score, with values of 0.970, 0.970, and 0.970, respectively. EfficientNetB0 obtained a macro F1-score of 0.953, while NasNetMobile obtained 0.908; these results indicate that MobileNetV2 provided the strongest overall classification performance among the three lightweight architectures evaluated under the predefined hold-out evaluation.

Although EfficientNetB0 obtained the highest validation accuracy at 95.65%, its test accuracy was lower than that of MobileNetV2. Therefore, validation accuracy alone was not used to determine the best-performing architecture. The subsequent analysis considers the evidence for cross-validation stability available and deployment-related characteristics to provide a broader assessment of the suitability of lightweight CNN for Android environments.

Table 4: Five-fold cross-validation results for MobileNetV2

Model	Validation Accuracy	Test Accuracy	Macro Precision	Macro Recall	Macro F1-score
MobileNetV2	93.84%	97.10%	0.970	0.970	0.970
MobileNetV2	93.48%	91.30%	0.910	0.918	0.908
EfficientNetB0	95.65%	95.65%	0.955	0.953	0.953

3.3 Model Complexity and Deployment Considerations

The complexity of the model is relevant to mobile deployment because predictive performance alone does not determine the computational suitability of a CNN architecture. In this study, MobileNetV2, NasNetMobile, and EfficientNetB0 were selected as lightweight candidate architectures based on their intended use in resource-constrained environments. However, quantitative measurements of total parameters, trainable parameters, non-trainable parameters, and model size were not available in the documented experimental records used for this revision. Therefore, these metrics are not reported as measured experimental results.

Accordingly, the model-selection analysis does not claim a quantitative superiority of one architecture over another in terms of model size or computational complexity. Instead, the available evidence emphasizes hold-out classification performance, the documented five-fold stability analysis for MobileNetV2, and successful functional Android deployment. A systematic parameter-count, model-size, and on-device efficiency benchmark is identified as an important direction for future work.

3.4 TensorFlow Lite Conversion Results

The trained lightweight CNN models were converted to TensorFlow Lite (TFLite) format to assess their suitability for Android deployment. The conversion was performed using the TensorFlow Lite Converter without applying additional post-training quantization or compression techniques. Therefore, the resulting TFLite models represent converted floating-point models rather than quantized versions INT8 or float16. The conversion step was intended to provide a deployable model representation for the Android application while preserving the classification behavior of the original trained models.

The TFLite conversion results are considered separately from model classification performance. In particular, conversion to TFLite does not by itself demonstrate improved accuracy, reduced latency, or lower memory consumption. These deployment characteristics require measurements on the target Android hardware and are therefore discussed separately in subsection 3.5.

Table 5: TensorFlow lite conversion configuration

Model	Conversion Format	Quantization)	Conversion Status
MobileNetV2	TensorFlow/Keras → TFLite	None	Converted
NasNetMobile	TensorFlow/Keras → TFLite	None	Converted
EfficientNetB0	TensorFlow/Keras → TFLite	None	Converted

3.5 Android On-Device Performance

The selected MobileNetV2 model was integrated into an Android application to evaluate its functional deployment on mobile devices. The application supports image input from the camera or gallery, preprocessing, CNN inference, and presentation of the predicted disease class and confidence score. Functional deployment was evaluated on two Android smartphones, namely the Infinix Smart 6 and Redmi Note 12, representing different hardware configurations.

For mobile classification testing, 30 previously unseen images were used, consisting of five images for each of the six classes. The images were not used during the model training and hold-out evaluation. The application correctly classified 29 of the 30 images, corresponding to an accuracy of 96.67%. One Healthy sample was incorrectly classified as another class. This result demonstrates that the trained model can be executed within the Android application and produce classification outputs on previously unseen mobile test images.

The mobile evaluation in this study is primarily a functional deployment assessment. Quantitative measurements of CPU latency, NNAPI latency, RAM consumption, and thermal behavior were not available in the documented experimental records and are therefore not reported as measured results. Consequently, the mobile accuracy of 96.67% should not be interpreted as a comprehensive benchmark of the efficiency of the inference of Android.

Table 6: Android mobile classification results

Class	Images Tested	Correct)	Misclassified	Accuracy
Leaf Spot	5	5	0	100%
Downy Mildew	5	5	0	100%
Blight	5	5	0	100%
Rust	5	5	0	100%
Healthy	5	4	1	80%
Non-leaf	5	5	0	100%
Total	30	29	1	96.67%

3.6 Model Selection Based on Classification and Deployment Evidence

The results of the hold-out evaluation and the five-fold cross-validation provide complementary evidence to select a lightweight CNN architecture for the intended Android application. Among the three models evaluated, MobileNetV2 achieved the highest hold-out test accuracy (97.10%) and the highest macro F1-score (0.970). Its five-fold cross-validation also demonstrated relatively stable validation performance, with a mean accuracy of 95.51% ± 0.92%. These results indicate that MobileNetV2 provided the strongest classification performance among the evaluated architectures, with additional evidence of validation stability from the available cross-validation analysis.

EfficientNetB0 achieved a higher validation accuracy (95.65%) than NasNetMobile (93.48%), but its test accuracy and macro F1-score remained below those of MobileNetV2. NasNetMobile showed the lowest overall hold-out classification performance, with a test accuracy of 91.30% and macro F1-score of 0.908. Therefore, based on the available classi-

fication evidence, MobileNetV2 was selected as the preferred lightweight architecture for the Android application.

The Android evaluation also demonstrated the practical feasibility of deploying the selected model in a mobile application. The application correctly classified 29 of 30 previously unseen images, corresponding to an accuracy of 96.67%. However, because quantitative measurements of CPU latency, NNAPI latency, RAM consumption, and thermal behavior were not available in documented experiments, the present study does not claim that MobileNetV2 is quantitatively the most efficient architecture on Android hardware. Instead, selection is based primarily on classification performance, cross-validation stability, and successful functional deployment.

Table 7: Overall performance and deployment evidence

Model	Test Accuracy	Macro F1-score)	5-Fold CV	Android Functional Test
MobileNetV2	97.10%	0.970	95.51 ± 0.92%	29/30 (96.67%)
NasNetMobile	91.30%	0.908	Not available	Not evaluated
EfficientNetB0	95.65%	0.953	Not available	Not evaluated

In general, MobileNetV2 demonstrated the strongest combination of classification performance, available cross-validation stability evidence, and successful Android functional deployment. These results support its selection as the preferred lightweight architecture for the proposed application. Nevertheless, a complete quantitative comparison of mobile efficiency among the three architectures would require additional measurements of model size, inference latency, memory consumption, and hardware-specific execution, which are beyond the documented experiments and are therefore identified as directions for future work.

4 Discussion

The results demonstrate that lightweight CNN architecture selection for mobile maize leaf disease classification should consider both predictive performance and practical deployment requirements. Among the three evaluated architectures, MobileNetV2 achieved the highest hold-out test accuracy of 97.10% and the highest macro F1-score of 0.970. NasNetMobile achieved a test accuracy of 91.30% with a macro F1-score of 0.908, while EfficientNetB0 achieved 95.65% test accuracy and a macro F1-score of 0.953. These results indicate that MobileNetV2 provided the strongest classification performance among the evaluated architectures under the documented hold-out evaluation.

The five-fold cross-validation results provide additional evidence regarding the stability of MobileNetV2. Its validation accuracy across the five folds ranged from 94.57% to 96.92%, with a mean of 95.51% and a standard deviation of 0.92 percentage points. The relatively small variation indicates that the observed MobileNetV2 performance was reasonably stable across different stratified partitions. However, because the documented cross-validation procedure used the complete dataset, including images belonging to the predefined hold-out subsets, the cross-validation result is interpreted as a stability analysis rather than an independent estimate of generalization.

The results also illustrate the importance of distinguishing classification performance from deployment efficiency. Although MobileNetV2 achieved the strongest classification performance in the available hold-out evaluation, this result alone does not establish that it is the most computationally efficient architecture on Android hardware. Model size, inference latency, memory consumption, and hardware-specific execution are important considerations for practical deployment. In the present study, the available deployment evidence primarily concerns functional execution of the classification pipeline rather than a complete quantitative hardware benchmark.

The MobileNetV2-based application was tested on an Infinix Smart 6 and a Redmi Note 12, and a separate evaluation using 30 previously unseen images produced 29 correct predictions, corresponding to an observed accuracy of 96.67%. Five images were evaluated for each of the six classes. Therefore, this result provides evidence of functional mobile deployment, but it should not be interpreted as a statistically robust estimate of field-level generalization.

Overall, the findings support MobileNetV2 as the preferred architecture for the present application within the documented experimental evidence because it combines the highest hold-out classification performance with available evidence of stable cross-validation performance and successful functional Android deployment. Nevertheless, the absence of complete quantitative measurements of CPU latency, NNAPI latency, memory consumption, and model size limits the extent to which computational efficiency can be compared across the three architectures. Future work should therefore include systematic on-device benchmarking and post-training optimization, such as float16 or INT8 quantization, to further investigate the accuracy–efficiency trade-off for Android deployment.

5 Conclusion

This study evaluated three lightweight CNN architectures MobileNetV2, NasNetMobile, and EfficientNetB0 for maize leaf disease classification with functional Android deployment. Based on the documented hold-out evaluation, MobileNetV2 achieved the strongest classification performance, with a test accuracy of 97.10%, macro precision of 0.970, macro recall of 0.970, and macro F1-score of 0.970. NasNetMobile achieved a test accuracy of 91.30% and macro F1-score of 0.908, while EfficientNetB0 achieved 95.65% test accuracy and a macro F1-score of 0.953.

The five-fold cross-validation analysis of MobileNetV2 produced a mean validation accuracy of 95.51% ± 0.92%, indicating relatively stable performance across the evaluated data partitions. Because the cross-validation procedure used the complete dataset, including images from the predefined hold-out subsets, this result is interpreted as evidence of performance stability rather than as an independent estimate of generalization.

The selected classification workflow was successfully integrated into an Android application and functionally evaluated on smartphone devices. The mobile evaluation produced 29 correct predictions out of 30 previously unseen images, corresponding to an observed accuracy of 96.67%. This result demonstrates the feasibility of executing the classification workflow on Android devices, although the small mobile test set limits its interpretation as evidence of field-level generalization.

Overall, the findings support MobileNetV2 as the preferred architecture for the present application within the documented experimental evidence because it combines the high-

est hold-out classification performance with available evidence of stable cross-validation performance and successful functional Android deployment. However, the present study does not provide a complete quantitative benchmark of mobile efficiency because systematic measurements of CPU latency, NNAPI latency, RAM consumption, and model size were not available in the documented experiments. Future work should therefore focus on systematic on-device benchmarking and post-training optimization, including float16 or INT8 quantization, to further investigate the accuracy–efficiency trade-off for practical Android deployment.

Acknowledgments

The authors would like to acknowledge the Doctoral Program in Information Systems, Graduate School, Universitas Diponegoro, Indonesia, and Politeknik Negeri Semarang, Indonesia, for the academic and institutional support provided during the research and development of this study.

References

[1] M. Reda, R. Suwwan, S. Alkafri, Y. Rashed, and T. Shanableh, "Agroaid: A mobile app system for visual classification of plant species and diseases using deep learning and tensorflow lite," in *Informatics*, vol. 9, p. 55, MDPI, 2022.

[2] İ. Avcı, M. Koca, and Y. Z. Khan, "A lightweight mobile deep learning framework for real-time plant disease detection in smart agriculture," in *2025 9th International Symposium on Innovative Approaches in Smart Technologies (ISAS)*, pp. 1–9, IEEE, 2025.

[3] L. J. L. Braz, J. L. de Moraes, and A. R. de Alexandria, "A lightweight deep learning model for accurate plant disease detection in real applications," in *2025 IEEE Latin Conference on IoT (LCIoT)*, pp. 94–97, IEEE, 2025.

[4] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 4510–4520, Ieee, 2018.

[5] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, pp. 6105–6114, PmLR, 2019.

[6] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 8697–8710, IEEE, 2018.

[7] M. Fraiwan, E. Faouri, and N. Khasawneh, "Classification of corn diseases from leaf images using deep transfer learning," *Plants*, vol. 11, no. 20, p. 2668, 2022.

[8] D. A. Padilla, R. A. I. Pajes, and J. T. De Guzman, "Detection of corn leaf diseases using convolutional neural network with openmp implementation," in *2020 IEEE 12th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, pp. 1–6, IEEE, 2020.

[9] F. Khan, N. Zafar, M. N. Tahir, M. Aqib, H. Waheed, and Z. Haroon, "A mobile-based system for maize plant leaf disease detection and classification using deep learning," *Frontiers in plant science*, vol. 14, p. 1079366, 2023.

[10] K. Ahadian, N. Yudistira, B. Rahayudi, A. H. Basori, S. J. Malebary, S. Alesawi, A. B. F. Mansur, A. S. Alorfi, and O. M. Barukab, "Maize disease classification using transfer learning and convolutional neural network with weighted loss," *Heliyon*, vol. 10, no. 21, 2024.

[11] F. Nan, Y. Song, X. Yu, C. Nie, Y. Liu, Y. Bai, D. Zou, C. Wang, D. Yin, W. Yang, *et al.*, "A novel method for maize leaf disease classification using the rgb-d post-segmentation image data," *Frontiers in Plant Science*, vol. 14, p. 1268015, 2023.

[12] J. Cai, R. Pan, J. Lin, J. Liu, L. Zhang, X. Wen, X. Chen, and X. Zhang, "Improved efficientnet for corn disease identification," *Frontiers in Plant Science*, vol. 14, p. 1224385, 2023.

[13] A. Gao, A. Geng, Y. Song, L. Ren, Y. Zhang, and X. Han, "Detection of maize leaf diseases using improved mobilenet v3-small," *International Journal of Agricultural and Biological Engineering*, vol. 16, no. 3, pp. 225–232, 2023.

[14] H. Zhou, Y. Su, J. Chen, J. Li, L. Ma, X. Liu, S. Lu, and Q. Wu, "Maize leaf disease recognition based on improved convolutional neural network shufflenetv2," *Plants*, vol. 13, no. 12, p. 1621, 2024.

[15] Z. Ji, S. Bao, M. Chen, and L. Wei, "Ics-resnet: a lightweight network for maize leaf disease classification," *Agronomy*, vol. 14, no. 7, p. 1587, 2024.

[16] I. Khan, S. S. Sohail, D. Ø. Madsen, and B. K. Khare, "Deep transfer learning for fine-grained maize leaf disease classification," *Journal of Agriculture and Food Research*, vol. 16, p. 101148, 2024.

[17] R. Li, Y. Li, W. Qin, A. Abbas, S. Li, R. Ji, Y. Wu, Y. He, and J. Yang, "Lightweight network for corn leaf disease identification based on improved yolo v8s," *Agriculture*, vol. 14, no. 2, p. 220, 2024.

[18] D. A. N. Gookyi, F. A. Wulnye, E. A. E. Arthur, R. K. Ahiadormey, J. O. Agyemang, K. O.-B. O. Agyekum, and R. Gyaang, "Tinyml for smart agriculture: Comparative analysis of tinyml platforms and practical deployment for maize leaf disease identification," *Smart Agricultural Technology*, vol. 8, p. 100490, 2024.

[19] E. O. Asani, Y. P. Osadeyi, A. A. Adegun, S. Viriri, J. A. Ayoola, and E. A. Kolawole, "mpd-app: a mobile-enabled plant diseases diagnosis application using convolutional neural network toward the attainment of a food secure world," *Frontiers in Artificial Intelligence*, vol. 6, p. 1227950, 2023.

[20] H. Ibrahim, A. M. Ahmed, B. Bouallegue, M. M. Khattab, M. A. El-Fattah, and N. A. El-Mawla, "Deep convolutional neural networks for plant disease detection: a mobile application approach (agri bot)," *International Journal of Intelligent Systems*, vol. 2025, no. 1, p. 7644407, 2025.

[21] J. Budiasto, H. Jayawardana, T. K. Rahayu, and N. P. Putra, "Klasifikasi penyakit daun jagung menggunakan mobilenetv2 dan tensorflow lite untuk deployment mobile di distrik semangga merauke," *MUSTEK ANIM HA*, vol. 15, no. 01, pp. 26–34, 2026.

[22] C. V. Padeiro, T.-W. Chen, T. Komamizu, and I. Ide, "Lightweight maize disease detection through post-training quantization with similarity preservation," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 2111–2120, IEEE, 2024.

[23] D. Singh, N. Jain, P. Jain, P. Kayal, S. Kumawat, and N. Batra, "Plantdoc: A dataset for visual plant disease detection," in *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*, pp. 249–253, 2020.

[24] V. S. Dhaka, S. V. Meena, G. Rani, D. Sinwar, M. F. Ijaz, and M. Woźniak, "A survey of deep convolutional neural networks applied for prediction of plant leaf diseases," *Sensors*, vol. 21, no. 14, p. 4749, 2021.

[25] J. Chen, J. Chen, D. Zhang, Y. Sun, and Y. A. Nanehkaran, "Using deep transfer learning for image-based plant disease identification," *Computers and electronics in agriculture*, vol. 173, p. 105393, 2020.

[26] S. M. Hassan, A. K. Maji, M. Jasiński, Z. Leonowicz, and E. Jasińska, "Identification of plant-leaf diseases using cnn and transfer-learning approach," *Electronics*, vol. 10, no. 12, p. 1388, 2021.

[27] M. Abubakar, Y. Ibrahim, O.-O. Ajayi, and S. S. Saminu, "A lightweight maize leaf disease recognition using pca-compressed mobilenetv2 features and rbf-svm," *Journal of Computing Theories and Applications*, vol. 3, no. 3, pp. 334–348, 2026.