Development of a Mobile Solution for Offline Disease Diagnosis in Moringa Leaves Using Convolutional Neural Networks

Maimun Huja Husin*,and Siti Hawa Aisyah Mohd Noor
Faculty of Engineering, Universiti Malaysia Sarawak, 94300 Kota Samarahan, Sarawak, Malaysia

Abstract

Mobile-based disease detection has become an important tool for crop health monitoring in resource-limited farming settings. Moringa oleifera is a fast-growing, drought-resistant, and highly nutritious tree that is nonetheless vulnerable to leaf diseases such as leaf spot, bacterial blight, powdery mildew, insect damage, rust and nutritional deficiency, yet very few diagnostic tools exist for this underutilised crop. This study presents the development of an offline mobile expert system for Moringa leaf disease diagnosis that combines convolutional neural networks (CNNs) with cross-platform mobile deployment. A total of 110 Moringa leaf images were collected, of which 102 images spanning six disease categories and a healthy class were pre-processed in MATLAB using resizing, Contrast Limited Adaptive Histogram Equalisation, Gaussian blur and RGB normalisation, then used to train and compare six pretrained CNN architectures—SqueezeNet, MobileNetV2, ResNet-18, ResNet-50, ResNet-101 and DenseNet-201—against accuracy, training time and model size. SqueezeNet reached the highest hold-out validation accuracy (52.63%, shared with MobileNetV2 and ResNet-101) with the shortest training time and the smallest model, around 95% smaller than ResNet-50, and was therefore selected for deployment. When the trained models were applied to all 102 dataset images, which include the training images, SqueezeNet, MobileNetV2 and ResNet-18 achieved 89.22%, 82.35% and 90.20%, respectively. Retaining RGB colour information improved accuracy by more than 10% relative to greyscale input. SqueezeNet was converted to ONNX and then to TensorFlow Lite, and embedded in a Flutter-based Android application designed for fully offline inference together with a local SQLite disease-and-treatment knowledge base. Mobile-integration testing confirmed offline image selection and on-device processing but also revealed implementation challenges, including intermittent inference failures linked to image-tensor handling in the Flutter runtime. The results demonstrate that lightweight CNNs can be practically deployed for offline plant disease diagnosis and support the broader goal of delivering accessible, data-driven agricultural tools to smallholder and rural Moringa farmers.

Keywords: *Moringa oleifera, leaf disease detection, convolutional neural network*

1. Introduction

Moringa oleifera is a fast-growing, drought-tolerant tree cultivated widely across tropical and subtropical regions, valued for its rich content of vitamins, minerals, protein and bioactive antioxidant compounds [1], [2]. In many developing countries, Moringa is an important source of low-cost nutrition and traditional medicine, yet like other crops it remains vulnerable to fungal, bacterial and

* Corresponding author.
E-mail address: hhmaimun@unimas.my
Manuscript History:
Received 14 August, 2026, Revised 27 August, 2026, Accepted 29 October, 2026, Published 31 October, 2026
Copyright © 2026 UNIMAS Publisher. This is an open access article under the CC BY-NC-SA 4.0 license.
https://doi.org/10.33736/jese.14055.2026

e-ISSN: 2838-0010

physiological diseases—including leaf spot, bacterial blight and powdery mildew—that can significantly reduce yield and nutritional quality if not detected early [3], [17]. Traditional disease diagnosis relies on the judgement of agricultural experts, a process that is time-consuming, labour-intensive and frequently inaccessible to farmers in rural or remote areas.

Mobile- and AI-based disease detection tools have accelerated crop health monitoring in several developing regions, giving farmers in countries such as India, Kenya and Indonesia real-time diagnostic support without requiring expert consultation [4]. However, such tools remain concentrated on staple crops such as rice and wheat, with comparatively little attention paid to underutilised but nutritionally important species like Moringa [1]. Many existing plant-disease detection systems also depend on cloud connectivity or computationally demanding models, which limits their applicability in low-connectivity rural settings where dependable internet access cannot be assumed [5].

This study addresses this gap by developing a mobile expert system that performs Moringa leaf disease diagnosis entirely offline. The specific objectives are to: (1) collect, pre-process and label a dataset of healthy and diseased Moringa leaf images; (2) train and comparatively evaluate several pretrained convolutional neural network (CNN) architectures for disease classification, selecting the most suitable model for mobile deployment; and (3) design and implement a cross-platform mobile application that embeds the trained model using TensorFlow Lite (TFLite) to deliver real-time, offline disease diagnosis together with locally stored treatment guidance. The resulting prototype targets farmers with limited technical background and unreliable internet connectivity, and is intended as a practical step toward more accessible, AI-assisted crop health management for sustainable Moringa cultivation.

Image-based plant disease detection has been studied extensively for staple and horticultural crops. Convolutional and edge-computing approaches have achieved strong results for leaf-level classification tasks in rice, banana and general leaf-disease datasets [6], [7], [13]–[16], while hyperspectral and IoT-based sensing methods offer complementary, non-visual routes to disease and stress detection at the cost of specialised, often expensive hardware [8], [9], [11], [12]. Comparative studies of multiclass leaf-disease classifiers further show that classical image-processing pipelines combined with machine learning can achieve reasonable accuracy but are typically less robust to real-world variation than deep CNN-based approaches [10], [18]. Collectively, this body of work establishes CNN-based image classification as a mature and effective approach for plant disease diagnosis; however, its application to Moringa, and its integration into a fully offline mobile pipeline suited to rural deployment, has received little dedicated attention, motivating the present study.

2. Materials and Methods

2.1 Dataset collection and preparation

A total of 110 RGB images of Moringa leaves were collected under natural lighting. Of these, 102 images were used for model development and organised into class-labelled folders comprising healthy leaves and six disease/condition categories: leaf spot, powdery mildew, bacterial blight, insect damage, rust disease and nutritional deficiency. The number of images used in each class is given in Table 1. Images were standardised for model input using MATLAB-based pre-processing: resizing to 224×224 pixels (227×227 for SqueezeNet) while preserving the RGB colour channels, and organisation via imageDatastore with automated batch-processing scripts. The dataset was partitioned using MATLAB's splitEachLabel function into training and validation subsets (approximately 80:20), which gave a hold-out validation subset of 19 images, and augmentation techniques—rotation, scaling and translation—were applied to the training data to improve robustness against real-world variability in leaf orientation, lighting and scale. The dataset is imbalanced, with only two bacterial-blight and six nutritional-deficiency images.

Table 1. Number of Moringa leaf images used in each class

Class	Number of images
Leaf spot	20
Powdery mildew	11
Insect damage	11
Bacterial blight	2
Rust disease	11
Nutritional deficiency	6
Healthy	41
Total used	**102**
Total collected	110

2.2. Image pre-processing

Prior to network input, each image underwent a fixed pre-processing pipeline to standardise size, contrast and colour distribution. Contrast Limited Adaptive Histogram Equalisation (CLAHE) was applied to enhance local contrast, particularly for images captured in low light; Gaussian blur ($\sigma = 1$) reduced high-frequency noise while preserving edge structure for downstream feature extraction; and RGB normalisation equalised pixel intensity distributions across the dataset. Table 2 summarises the pre-processing pipeline and its purpose.

Table 2. Image pre-processing pipeline

Pre-processing step	Tool	Purpose
Resize to 224×224 (227×227 for SqueezeNet)	MATLAB	Standardise CNN input size
CLAHE	MATLAB	Enhance local contrast
Gaussian blur ($\sigma = 1$)	MATLAB	Reduce noise, preserve edges
RGB normalisation	MATLAB	Equalise pixel intensity distribution

2.3. CNN architectures and training

Six pretrained CNN architectures were trained and comparatively evaluated: SqueezeNet, MobileNetV2, ResNet-18, ResNet-50, ResNet-101 and DenseNet-201. Each model's final fully connected and classification layers were replaced to match the number of target disease categories, and training was performed in MATLAB R2023a using the trainNetwork function over 50 epochs, with an initial learning rate of 0.0001 and a mini-batch size of 8. SqueezeNet was of particular interest owing to its lightweight 'fire module' architecture (alternating squeeze and expand convolutions), which substantially reduces parameter count relative to conventional CNNs while retaining competitive accuracy, making it well suited to mobile and embedded deployment. MobileNetV2 was included as a comparably mobile-oriented architecture using depthwise separable convolutions, while the ResNet and DenseNet variants served as deeper, higher-capacity baselines to quantify the accuracy-versus-efficiency trade-off relevant to offline mobile deployment.

Model performance was quantified using standard classification metrics. Accuracy, precision, recall and the F1-score were computed as shown in Equations (1)-(4), where TP, TN, FP and FN denote true positive, true negative, false positive and false negative counts, respectively. A model size

efficiency metric, Equation (5), was additionally used to express the storage-size reduction of a lightweight model relative to a baseline architecture.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

$$\text{Size efficiency} = \left(1 - \frac{S_{\text{SqueezeNet}}}{S_{\text{baseline}}}\right) \times 100\% \tag{5}$$

where $S_{\text{SqueezeNet}}$ and S_{baseline} denote the model sizes of SqueezeNet and the baseline architecture, respectively.

2.4. Model conversion and mobile application development

After training and validation in MATLAB, the selected model was exported using the exportONNXNetwork function to the Open Neural Network Exchange (ONNX) format and subsequently converted to TensorFlow Lite (.tflite), with post-training quantisation applied to reduce model size and accelerate inference on resource-constrained hardware. The mobile application was developed in Flutter, using Android Studio, and integrated with the trained model via the tflite_flutter plugin and the image_picker plugin for image selection. Figure 1 illustrates the overall workflow, which comprises a model-development stage and an on-device diagnosis stage. In the development stage (Figure 1a), the Moringa leaf images were pre-processed and used to train six pretrained CNNs in MATLAB R2023a; the selected SqueezeNet model was exported to ONNX and converted to TensorFlow Lite with post-training quantisation for integration into the Flutter application. In the diagnosis stage (Figure 1b), the user selects a leaf image from the phone gallery through the image_picker plugin; the image is resized to 224×224 pixels and normalised, and is processed by the embedded SqueezeNet model through the TensorFlow Lite interpreter. The predicted disease class is linked to the disease descriptions and recommended treatments stored in a local SQLite database, and the results are displayed to the user and saved locally. All steps of the diagnosis stage run on the device, so no Internet connection is required. The backend API, database, analytics, tracking service and external service of the initial concept design are Internet-dependent and are not part of the offline prototype; they are shown outside the offline boundary in Figure 1. Because Apple's iOS provisioning and code-signing requirements necessitate macOS-based development hardware that was unavailable for this project, the application was developed and tested for Android devices only.

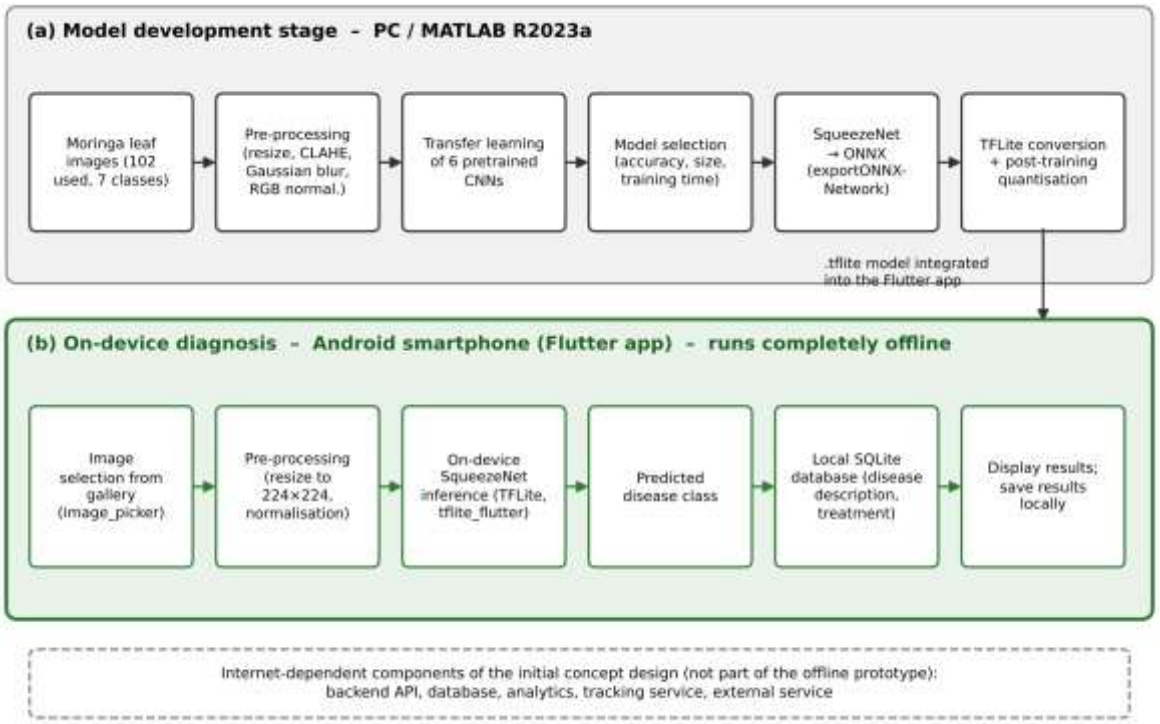


Figure 1. Overall system workflow: (a) model-development stage in MATLAB; (b) on-device diagnosis stage, which runs completely offline on the Android smartphone. The dashed box lists the Internet-dependent components of the initial concept design, which are not used by the offline prototype.

2.5. Performance evaluation

In addition to the accuracy, precision, recall and F1-score metrics described above, confusion matrices were generated for the three best-performing architectures—SqueezeNet, MobileNetV2 and ResNet-18—using MATLAB's confusionchart() function, with row- and column-normalisation applied to reveal per-class precision and recall. The validation accuracy in Table 3 was measured during training on the hold-out validation subset (19 images withheld from training). The confusion matrices, in contrast, were computed in a separate step by applying each saved model to all 102 dataset images, which include the images used for training; they therefore describe how well each model fits the collected images rather than its performance on unseen images. This analysis was used both to select the deployment model and to identify systematic sources of misclassification, such as visually similar disease symptoms.

3. Results and discussion

3.1. Effect of colour information on classification accuracy

An initial comparison examined the effect of retaining colour information on classification performance. Using RGB input, SqueezeNet achieved a validation accuracy of 52.63% on the hold-out validation subset, whereas converting the same images to greyscale reduced accuracy by more than 10 percentage points. This confirms that colour is a diagnostically important cue for Moringa leaf disease detection, since disease symptoms such as discolouration, spotting and mildew are frequently

distinguished by subtle chromatic variation that is lost in greyscale representations. Consequently, all subsequent training and deployment retained full RGB input.

3.2. Comparative CNN training performance

Table 3 summarises the validation accuracy, training time and relative model size obtained for the six CNN architectures trained on the RGB dataset. SqueezeNet and MobileNetV2 both reached a validation accuracy of 52.63%, but SqueezeNet required under 10 minutes of training against 36.35 minutes for MobileNetV2, and produced the smallest model footprint of the six architectures tested; according to Equation (5), SqueezeNet is around 95% smaller than ResNet-50. ResNet-101 matched the highest observed accuracy (52.63%) but required over 200 minutes to train and produced the largest model size, making it impractical for mobile deployment. ResNet-18 offered a reasonable balance of size and training time at a lower accuracy of 42.11%, while ResNet-50 and DenseNet-201 achieved intermediate accuracy (47.37%) at substantially higher computational cost, with DenseNet-201 alone requiring over six hours of training. Because the validation subset contains only 19 images, each image corresponds to 5.26 percentage points, so the differences between the models amount to at most two images. These results illustrate the fundamental trade-off between representational capacity and deployment efficiency: deeper networks are not automatically better suited to an offline mobile use case, since their accuracy gains do not compensate for the increase in size, inference latency and battery consumption on resource-constrained devices.

Table 3. Comparative CNN training performance (RGB input; validation accuracy on the hold-out validation subset of 19 images)

Model	Input size	Validation accuracy (%)	Training time (min)	Model size	Remarks
SqueezeNet	227×227×3	52.63	10.0	Very small	Lightweight, slightly lower accuracy
MobileNetV2	224×224×3	52.63	36.35	Small	Mobile-optimised, balanced performance
ResNet-18	224×224×3	42.11	31.59	Moderate	Balanced accuracy and speed
ResNet-50	224×224×3	47.37	84.47	Large	Higher accuracy, longer training time
ResNet-101	224×224×3	52.63	203.03	Very large	Highest accuracy, not mobile-suitable
DenseNet-201	224×224×3	47.37	374.35	Large	Good performance, resource-intensive

Model size is a relative ranking of the trained MATLAB networks before conversion to ONNX or TensorFlow Lite, from very small (SqueezeNet) to very large (ResNet-101); SqueezeNet is around 95% smaller than ResNet-50. Exact parameter counts and file sizes were not recorded in this study.

3.3. Confusion matrix evaluation

Following selection of the leading candidates, SqueezeNet, MobileNetV2 and ResNet-18 were further evaluated with per-class confusion matrices computed on all 102 dataset images. SqueezeNet achieved an overall classification accuracy of 89.22%, MobileNetV2 achieved 82.35%, and ResNet-18 achieved the highest accuracy of 90.20%. Because these 102 images include the images used for training, these values are higher than the hold-out validation accuracies in Table 3 and reflect how well the models fit the collected dataset rather than their accuracy on unseen images. The 52.63% and 89.22% values for SqueezeNet therefore come from two different evaluations of the same trained

model, not from two different datasets. Figure 2 shows the SqueezeNet confusion matrix, which shows strong diagonal dominance across all seven classes; the majority of misclassifications occurred between visually similar categories, such as leaf spot and rust disease, and between a small number of healthy samples and early-stage disease symptoms. MobileNetV2 showed a comparatively higher tendency to misclassify healthy leaves as diseased and to confuse powdery mildew with bacterial blight, consistent with its lighter-weight design trading some discriminative capacity for computational efficiency. ResNet-18 benefited from its deeper feature extraction to reduce misclassification overall, though it occasionally confused rust disease with leaf spot due to overlapping texture patterns. Given that SqueezeNet achieved accuracy within one percentage point of ResNet-18 at a fraction of the model size and training cost, and shared the highest hold-out validation accuracy, it was selected as the deployment model for the mobile application.

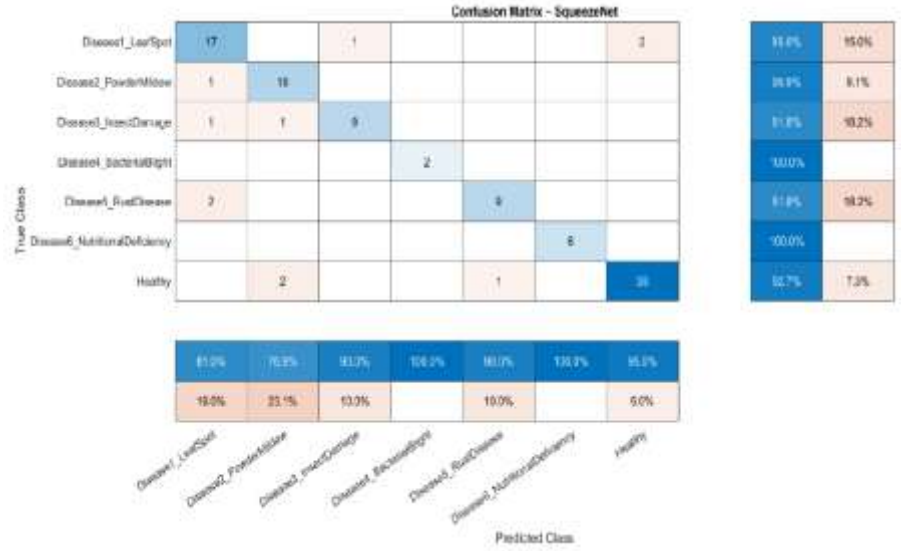


Figure 2. SqueezeNet confusion matrix computed on all 102 dataset images (row-normalised recall shown at right, column-normalised precision shown below).

Table 4 lists the per-class recall and precision obtained for SqueezeNet. Recall exceeded 80% for every class and reached 100% for bacterial blight and nutritional deficiency, the two most visually distinctive categories, while precision was lowest for powdery mildew (76.9%) owing to confusion with leaf spot and rust disease. The healthy class achieved both high recall (92.7%) and high precision (95.0%), indicating that the model rarely confused healthy foliage with diseased foliage overall, even though a small number of individual misclassifications did occur in both directions. The result for bacterial blight is based on only two images and should be interpreted with caution.

Table 4. Per-class recall and precision for SqueezeNet on all 102 dataset images

Class	Recall (%)	Precision (%)
Leaf spot	85.0	81.0
Powdery mildew	90.9	76.9
Insect damage	81.8	90.0
Bacterial blight	100.0	100.0
Rust disease	81.8	90.0
Nutritional deficiency	100.0	100.0
Healthy	92.7	95.0

3.4. Mobile application development and testing

The trained SqueezeNet model was exported to ONNX and converted to TensorFlow Lite for integration into the Flutter application. Figure 3 shows the application on an Android smartphone. During testing, the application successfully allowed users to select a leaf image from the gallery and displayed it on the interface (Figure 3a–b); the image was resized to 224×224 pixels and passed to the on-device TFLite interpreter without requiring an internet connection, consistent with the project's central objective of offline operation. However, the prediction pipeline occasionally stalled on a "Processing…" state without returning a classification result (Figure 3c), and in the tested build the result field did not yet present the model's predicted disease class (Figure 3b shows the example output label). Debugging traced this behaviour to a GUID (Globally Unique Identifier) parsing conflict combined with inconsistent handling of input image metadata and tensor conversion within the Flutter/Dart asynchronous execution model, together with memory mismanagement and asynchronous delays in the Dart code. Mitigation steps included verifying that input images were correctly normalised and resized, confirming full quantisation and plugin-version compatibility between image_picker and tflite_flutter, and inserting debug logging to trace intermediate outputs; these measures reduced, but did not fully eliminate, the intermittent failures. The prototype nonetheless confirmed the technical feasibility of loading a trained CNN model and running the image-selection and inference pipeline offline on a standard Android smartphone, establishing a practical foundation for further engineering refinement.

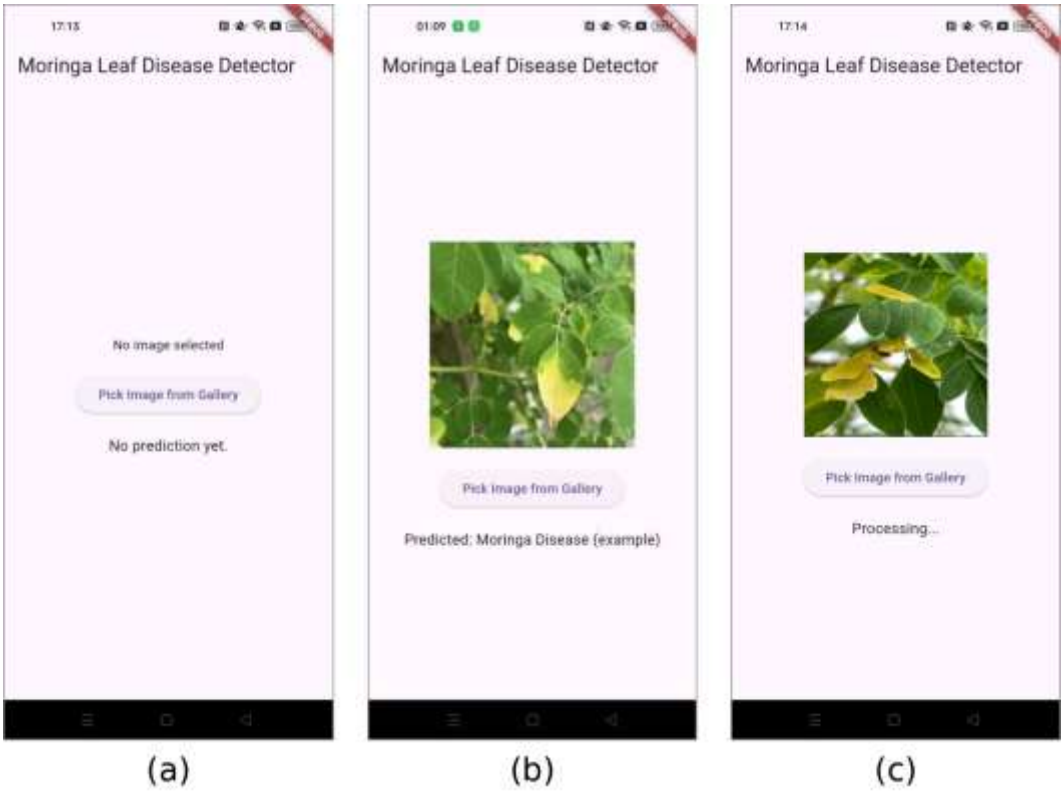


Figure 3. Screenshots of the Android application prototype: (a) main page before image selection; (b) Moringa leaf image (rust disease) selected from the gallery and displayed, with the prototype's example output label; (c) on-device processing state.

3.5. Discussion

The results highlight a two-stage trade-off inherent to offline mobile disease diagnosis. At the model-training stage, deeper architectures such as ResNet-101 and DenseNet-201 offered no decisive accuracy advantage over SqueezeNet on the hold-out validation subset, while incurring order-of-magnitude increases in training time and storage footprint that make them impractical for mid-range smartphones used in rural settings. At the deployment stage, the gap between MATLAB-based model validation and real-world mobile behaviour proved to be the more significant engineering challenge: robust, production-grade error handling around asynchronous inference and tensor conversion is at least as important as model selection in delivering a reliable end-user experience. These findings reinforce the broader case for lightweight CNN architectures in resource-limited agricultural applications, while underscoring that mobile-software engineering practices—rather than model accuracy alone—ultimately determine whether such systems are usable by non-technical, rural end users. By combining an accurate, compact classification model with an offline-capable mobile interface, this work contributes toward reducing farmers' dependence on expert consultation and supports Sustainable Development Goal 2 (Zero Hunger) and Goal 9 (Industry, Innovation and Infrastructure) through more accessible, data-driven crop health management.

3.6. Comparison with previous studies

Table 5 compares the present study with CNN-based plant-disease studies on other crops. Studies on rice, banana, maize and oil palm report accuracies of 88–95% using datasets of 1,200–2,000 images [13], [16], [19], [20], and only the banana study [16] provides offline capability. The validation accuracy of the present study (52.63%) is lower, which reflects its much smaller dataset of 102 images. However, it is the only one of these studies that targets Moringa and is designed to run fully offline on a smartphone, addressing a crop for which disease-detection tools are lacking. Expanding the Moringa dataset is therefore the main route to closing the accuracy gap while preserving offline operation.

Table 5. Comparison of CNN-based plant-disease detection studies

Study	Crop	Model	Accuracy (%)	Dataset size (images)	Offline capability
[13]	Rice	ResNet-50	93.1	2,000	No
[16]	Banana	MobileNetV2	95	1,500	Yes
[19]	Maize	InceptionV3	91	1,800	No
[20]	Oil palm	VGG16	88	1,200	No
This study	Moringa	SqueezeNet	52.63	102	Yes

3.7. Limitations

Several limitations should be considered when interpreting these results. The dataset, while sufficient to demonstrate feasibility, remained modest in size (110 images collected and 102 used across seven classes, with only two bacterial-blight images), which constrains the model's ability to generalise to disease presentations not represented in training, such as different growth stages, lighting conditions or geographic origins. The hold-out validation subset contained only 19 images, and the confusion matrices were computed on the full dataset including the training images, so the reported per-class results do not measure performance on unseen images. The mobile application was developed and validated on Android devices only, owing to the macOS-based hardware and provisioning requirements of iOS development, and was tested on a limited range of device models; inference latency and memory behaviour may therefore differ on lower-specification smartphones

common in rural settings. The device specifications, converted model size and on-device inference time were not recorded in this study. Finally, the intermittent inference failures observed during testing indicate that further software hardening is required before the prototype could be considered production-ready for deployment to end users.

3.8. Ethical and social impact

Beyond its technical contribution, an offline diagnostic tool of this kind carries direct social value for smallholder and rural Moringa farmers who may otherwise lack access to agricultural extension expertise. By enabling early, low-cost disease identification without reliance on internet connectivity, the system can support more targeted treatment decisions—for example, reducing unnecessary pesticide application—thereby lowering production costs and environmental impact while helping to safeguard the nutritional and economic value of Moringa cultivation. These outcomes align with Sustainable Development Goal 2 (Zero Hunger), through improved crop-health management and food security, and Sustainable Development Goal 9 (Industry, Innovation and Infrastructure), through the extension of digital agricultural tools to underserved and technologically underserved farming communities.

4. Conclusion

This study developed and evaluated an offline mobile expert system for diagnosing Moringa leaf diseases using convolutional neural networks. A labelled dataset of 110 collected images, of which 102 were used, spanning six disease categories and a healthy class was used to train and compare six CNN architectures, from which SqueezeNet was selected for its favourable balance of validation accuracy (52.63%, the highest observed), smallest model size (around 95% smaller than ResNet-50) and shortest training time. When applied to all 102 dataset images, SqueezeNet achieved 89.22%, compared with 90.20% for the heavier ResNet-18 and 82.35% for MobileNetV2. Retaining RGB colour input rather than greyscale improved accuracy by over 10 percentage points, confirming the diagnostic value of colour information for this task. The trained model was successfully converted to TensorFlow Lite and integrated into a Flutter-based Android application designed for fully offline inference and locally stored treatment guidance, though intermittent inference failures linked to asynchronous tensor handling highlight the need for more robust mobile-software engineering. Future work should focus on expanding the training dataset with greater diversity in disease stage, lighting and location; strengthening error handling and input validation within the mobile application; and evaluating newer efficiency-oriented architectures such as EfficientNet-Lite, MobileNetV3 or TinyML frameworks to further improve the accuracy-to-efficiency ratio for offline agricultural diagnostic tools.

Acknowledgements

The authors thank the Faculty of Engineering, Universiti Malaysia Sarawak (UNIMAS), for the facilities and academic support provided during this study.

Conflicts of interest

We declare no conflict regarding the publication of the study.

References

[1] Alegbeleye, O. O. (2018). How functional is Moringa oleifera? A review of its nutritive, medicinal, and socioeconomic potential. *Food and Nutrition Bulletin, 39*(1), 149–170. https://doi.org/10.1177/0379572117749814

[2] Kashyap, P., Kumar, S., Riar, C. S., Jindal, N., Baniwal, P., Guiné, R. P. F., Correia, P. M. R., Mehra, R., & Kumar, H. (2022). Recent advances in drumstick (Moringa oleifera) leaves bioactive compounds: Composition, health benefits, bioaccessibility, and dietary applications. *Antioxidants, 11*(2), 402. https://doi.org/10.3390/antiox11020402

[3] Bartíková, M., Holková, L., Šafránková, I., & Němec, P. (2020). Causal agents of powdery mildew on Moringa stenopetala (Baker f.) Cuf. and Moringa oleifera Lam. in Ethiopia. *South African Journal of Botany, 129*, 457–462. https://doi.org/10.1016/j.sajb.2019.12.002

[4] Nair, R. R., Selvi, R. S., Vartini, G. H., Jayashri, B., Suruthi, S., & Anand, A. J. (2023). Maturity level classification of Moringa leaf using DNN algorithm. In *Proceedings of the 4th International Conference on Smart Electronics and Communication (ICOSEC 2023)* (pp. 1667–1672). IEEE. https://doi.org/10.1109/ICOSEC58147.2023.10276048

[5] Chakraborty, S., & Newton, A. C. (2011). Climate change, plant diseases and food security: An overview. *Plant Pathology, 60*(1), 2–14. https://doi.org/10.1111/j.1365-3059.2010.02411.x

[6] Sharma, K., Jayanthi, J., Kumar, A., Jain, S., Jain, G., & Vishwakarma, H. (2024). Computer vision based smart leaf disease detection system. In *2024 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS 2024)*. IEEE. https://doi.org/10.1109/SCEECS61402.2024.10481853

[7] Nikith, B. V., Keerthan, N. K. S., Praneeth, M. S., & Amrita, D. T. (2023). Leaf disease detection and classification. *Procedia Computer Science, 218*, 291–300. https://doi.org/10.1016/j.procs.2023.01.011

[8] Reis-Pereira, M., Tosin, R., Martins, R. C., Dos Santos, F. N., Tavares, F., & Cunha, M. (2023). Enhancing kiwi bacterial canker leaf assessment: Integrating hyperspectral-based vegetation indexes in predictive modeling. *Engineering Proceedings, 48*(1), 22. https://doi.org/10.3390/CSAC2023-14920

[9] Wan, L., Li, H., Li, C., Wang, A., Yang, Y., & Wang, P. (2022). Hyperspectral sensing of plant diseases: Principle and methods. *Agronomy, 12*(6), 1451. https://doi.org/10.3390/agronomy12061451

[10] Ganatra, N., & Patel, A. (2020). A multiclass plant leaf disease detection using image processing and machine learning techniques. *International Journal on Emerging Technologies, 11*(2), 388–421.

[11] Venkatachala Appa Swamy, M., Periyasamy, J., Thangavel, M., Khan, S. B., Almusharraf, A., Santhanam, P., Ramaraj, V., & Elsisi, M. (2023). Design and development of IoT and deep ensemble learning based model for disease monitoring and prediction. *Diagnostics, 13*(11), 1942. https://doi.org/10.3390/diagnostics13111942

[12] Rehman, A., Saba, T., Kashif, M., Fati, S. M., Bahaj, S. A., & Chaudhry, H. (2022). A revisit of internet of things technologies for monitoring and control strategies in smart agriculture. *Agronomy, 12*(1), 127. https://doi.org/10.3390/agronomy12010127

[13] Kumar, D. H., & Punniakodi, S. (2024). Rice leaf disease detection using ResNet50. In *2024 International Conference on Signal Processing, Computation, Electronics, Power and Telecommunication (IConSCEPT 2024)*. IEEE. https://doi.org/10.1109/IConSCEPT61884.2024.10627835

[14] Saputra, A. D., Hindarto, D., & Santoso, H. (2023). Disease classification on rice leaves using DenseNet121, DenseNet169, DenseNet201. *Sinkron, 8*(1), 48–55. https://doi.org/10.33395/sinkron.v8i1.11906

[15] Da Silva, P. E. C., & Almeida, J. (2025). An edge computing-based solution for real-time leaf disease classification using thermal imaging. *IEEE Geoscience and Remote Sensing Letters, 22*. https://doi.org/10.1109/LGRS.2024.3456637

[16] Haq, M. K., Tanvir, K., Kabir, M. S., Alam, N. S. B., Dey, S. K., & Raju, V. (2024). Enhancing banana leaf spot disease classification using dense MobileNetV2. In *Proceedings – 3rd International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI 2024)*. IEEE. https://doi.org/10.1109/ACCAI61061.2024.10602228

[17] Grace, M., Gatan, B., Joshi, R., Dale, P., Roman, O., & Gatan, M. T. (2020). *Diseases of Moringa and their management* [SSRN Working Paper No. 3582292]. https://doi.org/10.2139/ssrn.3582292

[18] Hasan, M. N., Mustavi, M., Jubaer, M. A., Shahriar, M. T., & Ahmed, T. (2022). Plant leaf disease detection using image processing: A comprehensive review. *Malaysian Journal of Science and Advanced Technology, 2*(4), 174–182. https://doi.org/10.56532/mjsat.v2i4.80

[19] Daqing, W., Lin, L., Xinglong, Y., Lili, G., & Zhipeng, R. (2023). Maize leaf diseases identification using EfficientNet based on deep transfer learning. *Journal of Northeast Agricultural University, 2023*(5), 66–76. https://doi.org/10.19720/j.cnki.issn.1005-9369.2023.05.008

[20] Kurihara, J., Koo, V. C., Guey, C. W., Lee, Y. P., & Abidin, H. (2022). Early detection of basal stem rot disease in oil palm tree using unmanned aerial vehicle-based hyperspectral imaging. *Remote Sensing, 14*(3), 799. https://doi.org/10.3390/rs14030799