Lightweight CNN Architectures for Real-Time Crop Disease Detection on Low-End Smartphones for Rural Deployment

Soham Sushil Parab
SY MSc Data Science, MIT Arts, Commerce and Science College
doi.org/10.64643/ijirt.208558-459

***Abstract*—Crop disease diagnosis remains a critical bottleneck for smallholder farmers, particularly in rural and semi-urban regions where agricultural extension services are limited and internet connectivity is unreliable. While convolutional neural networks (CNNs) have achieved high accuracy in plant disease classification, most existing research prioritizes accuracy alone, using large architectures such as ResNet and VGG that are impractical for deployment on low-end smartphones. This study addresses that gap by evaluating MobileNetV3-Small, EfficientNet-Lite0, and a custom pruned/quantized CNN against a ResNet50 baseline for multi-class crop disease classification using the PlantVillage dataset. Beyond accuracy and F1-score, models are benchmarked using model size, CPU-only inference latency, and memory footprint under conditions representative of budget Android devices with no GPU and offline operation. The best-performing lightweight model is further optimized using post-training quantization, and its accuracy-efficiency trade-off is analyzed before and after compression. The study proposes an accuracy-per-MB and accuracy-per-millisecond framework to compare deployment feasibility across architectures. Since verified experimental measurements were not available during document preparation, numerical results remain to be experimentally determined.**

***Index Terms*—crop disease detection, convolutional neural networks, lightweight CNNs, edge AI, mobile deployment.**

I. INTRODUCTION

Plant diseases threaten agricultural productivity and farmer income, while visual diagnosis can be slow, subjective, and dependent on scarce expertise. The PlantVillage initiative was motivated partly by the possibility of smartphone-assisted plant-disease diagnosis and released more than 50,000 curated leaf images for machine-learning research [1]. Mohanty, Hughes, and Salathé subsequently trained a deep convolutional neural network on 54,306 images representing 14 crop species and 26 diseases, reporting 99.35% held-out accuracy under controlled conditions [2]. These results demonstrate the potential of deep learning, but they do not by themselves establish field reliability or suitability for budget smartphones.

The deployment problem has two coupled dimensions. First, a classifier must correctly distinguish disease classes. Second, it must satisfy device constraints involving storage, RAM, CPU time, battery consumption, and offline execution. ResNet50 provides a strong accuracy-oriented baseline because residual learning enables the optimization of deep networks [3]. However, its computational cost motivates mobile-oriented architectures. MobileNetV3 combines hardware-aware neural architecture search with efficient mobile building blocks [4].

EfficientNet introduced compound scaling across depth, width, and resolution, while EfficientNet-Lite adapted this design family for TensorFlow Lite-oriented edge deployment [5], [6].

This study asks which of ResNet50, MobileNetV3-Small, EfficientNet-Lite0, and a custom pruned/quantized CNN provides the most useful balance between crop-disease classification quality and CPU-only deployment cost on low-end smartphones. The study objectives are to compare the models under a common protocol, quantify classification and deployment metrics, evaluate compression, and establish a reproducible offline benchmarking framework.

The main contribution is a deployment-aware comparison that combines classification quality with

model size, CPU latency, memory footprint, and two interpretable efficiency indicators. The study does not claim that either indicator is globally novel; rather, it proposes them as practical tools for this application.

II. RELATED WORK

A. Plant-Disease Classification

The PlantVillage dataset established an important open benchmark for plant-disease recognition [1]. Mohanty et al. demonstrated that transfer learning and deep CNNs can achieve very high in-domain performance on controlled leaf images [2]. However, the images are generally cleaner and less variable than images captured by farmers in actual fields.

PlantDoc was introduced to address this limitation. It contains 2,598 images across 13 plant species and up to 17 disease classes, with images collected from non-controlled environments [7]. Field images contain variable lighting, complex backgrounds, occlusion, multiple leaves, different camera qualities, and disease symptoms at different stages. Reviews identify dataset homogeneity, limited field validation, small lesion visibility, and insufficient generalization as continuing challenges [8].

B. Efficient CNN Architectures

ResNet50 is used as the baseline because residual connections allow deep networks to be trained effectively [3]. MobileNet architectures use depthwise separable convolutions to reduce computation relative to conventional convolutions. MobileNetV3 extends this family through hardware-aware search and mobile-specific design choices [4]. EfficientNet balances network depth, width, and input resolution through compound scaling [5]. EfficientNet-Lite adapts the EfficientNet family for mobile CPUs, GPUs, and EdgeTPU devices [6].

Architecture-level efficiency does not guarantee application-level efficiency. Actual performance also depends on the runtime, operator implementation, number of CPU threads, preprocessing, memory allocation, device thermal state, and operating-system overhead.

C. Compression and Edge Deployment

Pruning removes parameters, filters, channels, or other structures from a neural network. Structured pruning is especially relevant to mobile deployment because removing complete channels or filters can reduce actual tensor dimensions and operator costs [9]. Deep Compression combined pruning, trained quantization, and Huffman coding in a three-stage compression pipeline [10].

Post-training quantization converts a trained floating-point model into lower-precision representations. Full-integer INT8 quantization requires representative calibration data and may enable efficient integer execution on compatible hardware [11]. TensorFlow Lite benchmarking tools can report initialization time, steady-state inference latency, memory-related measurements, and operator-level timing [12].

The research gap is the limited availability of a reproducible, application-specific comparison that evaluates classification quality and deployment cost under the same data, preprocessing, runtime, and benchmarking protocol. Prior work often reports accuracy without CPU latency, memory, deployed size, or detailed hardware conditions. Controlled PlantVillage performance may also fail to transfer to field conditions.

III. METHODOLOGY

A. Dataset

PlantVillage contains approximately 54,303–54,306 images depending on the distribution and metadata version. The commonly cited formulation contains 38 classes representing 14 crop species and 26 diseases or healthy conditions [1], [2], [13]. The exact dataset version, file manifest, class names, image counts, and file hashes must be recorded.

Since multiple images may originate from the same leaf or image source, random image-level splitting can introduce leakage. Where metadata permits, splitting should be performed by leaf or source-image identity. If this is not possible, the limitation must be documented. PlantVillage is suitable for controlled architectural comparison but should not be treated as a complete representation of field conditions.

B. Data Preprocessing

Images should be decoded as RGB, resized to 224×224 pixels, and normalized according to each backbone's preprocessing convention. Training augmentation should include random horizontal flips, bounded rotation, random crop and scale, and moderate brightness, contrast, and color variation.

Validation and test preprocessing must remain deterministic.

A stratified 70/15/15 training-validation-test partition is proposed, using random seed 42. If the downloaded dataset has meaningful class imbalance, macro-averaged metrics, balanced sampling, or class-weighted loss should be used. Exact class counts should be reported from the final manifest.

C. Model Architectures

ResNet50 is the accuracy-oriented baseline. Its residual connections support deep network optimization [3]. The original final classifier is replaced with a 38-class classification layer.

MobileNetV3-Small is selected for resource-constrained mobile inference. Its depthwise operations and compact channel dimensions make it appropriate for low-end CPU deployment [4].

EfficientNet-Lite0 provides a comparison between MobileNet-style efficiency and compound-scaled feature extraction. The Lite family is designed for mobile and edge runtimes [6].

The custom CNN should contain a convolutional stem, depthwise-separable blocks, batch normalization, ReLU6 or another quantization-compatible activation, global average pooling, and a compact classifier. The exact blocks and channel widths must be released before training.

D. Training Procedure

TABLE I. PROPOSED TRAINING CONFIGURATION. THESE SETTINGS ARE NOT EXPERIMENTALLY VERIFIED RESULTS

Setting	Proposed value
Initialization	ImageNet transfer learning where available
Optimizer	AdamW or SGD, selected before test evaluation
Learning rates	1×10−3 for head; 1×10−4 for fine-tuning
Batch size	32
Maximum epochs	40
Loss	Categorical cross-entropy
Scheduler	Cosine decay
Early stopping	Patience of 7 epochs
Selection metric	Validation macro-F1
Random seed	42
Input size	224×224
Runtime	CPU-only TensorFlow Lite/LiteRT

E. Model Compression and Quantization

The compression workflow consists of training the floating-point model, evaluating it on the fixed test set, ranking filters or channels using a predefined importance criterion, applying structured channel pruning, fine-tuning, exporting to the mobile runtime, applying post-training quantization, and re-evaluating accuracy, size, latency, and memory.

The study should evaluate FP32, FP16-weight, and full-integer INT8 variants. The calibration set must be disjoint from the test set and should reflect the expected input distribution. Quantization must be validated for operator compatibility on the selected runtime [11].

F. Evaluation Metrics

For class c, precision, recall, and F1-score are defined as $P_c=TP_c/(TP_c+FP_c)$, $R_c=TP_c/(TP_c+FN_c)$, and $F1_c=2P_cR_c/(P_c+R_c)$. Overall accuracy is $A=\Sigma_cTP_c/N$. Macro-F1 is the unweighted mean of the class-level F1-scores.

The proposed efficiency indicators are $E_{MB}=A/S_{MB}$ and $E_{ms}=A/L_{ms}$, where A is accuracy, S is deployed model size in megabytes, and L is median CPU-only inference latency in milliseconds. These ratios are decision aids, not substitutes for class-wise safety or field validation.

G. Deployment Benchmarking

All models should be exported to the same runtime, with GPU and NNAPI delegates disabled for the CPU-only comparison. The device model, operating-system version, runtime version, thread count, battery level, thermal state, and background-app state should be recorded. After warm-up, at least 100 single-image inference runs should be executed. Median, mean, standard deviation, and p95 latency should be reported. The benchmark should be repeated at least three times. TensorFlow Lite tools support latency aggregation, memory reporting, and operator profiling [12].

IV. RESULTS AND DISCUSSION

No execution environment, target smartphone, downloaded dataset manifest, or trained model artifacts were available to this paper-generation process. Accordingly, experimental values are

intentionally not fabricated and remain "To be experimentally determined."

TABLE II. PLANNED CLASSIFICATION AND DEPLOYMENT RESULTS. TBD = TO BE EXPERIMENTALLY DETERMINED

Model	Acc.	Macro -F1	Size	CPU ms	Peak MB
ResNet50	TBD	TBD	TBD	TBD	TBD
MobileNetV3 -Small	TBD	TBD	TBD	TBD	TBD
EfficientNet-Lite0	TBD	TBD	TBD	TBD	TBD
Custom CNN FP32	TBD	TBD	TBD	TBD	TBD
Custom CNN INT8	TBD	TBD	TBD	TBD	TBD

After execution, the analysis should compare macro-F1 rather than accuracy alone, inspect confusion matrices for disease pairs, and report repeated-run variation where feasible. Quantization should be considered beneficial only if reductions in size and latency are accompanied by an acceptable change in task performance. A model should not be called "best" without defining whether best means highest F1, smallest footprint, lowest latency, or Pareto-efficient compromise.

Published PlantVillage results illustrate this caution. Mohanty et al. reported 99.35% accuracy on held-out controlled images [2], while later work has emphasized that models trained on controlled imagery may degrade when evaluated on field images [8], [14]. Thus, a strong PlantVillage result should be interpreted as benchmark evidence rather than proof of farmer-facing diagnostic reliability.

The central trade-off is multi-objective. A model with the highest classification accuracy may be impractical if it exceeds storage, memory, or latency limits. Conversely, a compact model may be unsuitable if its errors are concentrated in agronomically important disease classes.

For rural and offline use, a practical system should support local inference without cloud connectivity, confidence thresholds, an "uncertain" output, image-quality checks, local diagnostic history, optional synchronization, and human or agronomist escalation for high-risk cases.

V. CONCLUSION

This paper establishes a reproducible methodology for studying lightweight CNN-based crop-disease classification under rural smartphone deployment constraints. It preserves ResNet50 as the baseline and evaluates MobileNetV3-Small, EfficientNet-Lite0, and a custom pruned/quantized CNN using both classification and deployment-oriented measures.

The principal methodological contribution is a transparent evaluation framework combining accuracy, precision, recall, macro-F1, model size, CPU-only latency, memory footprint, and the proposed accuracy-per-MB and accuracy-per-millisecond indicators. Since the experiments were not executed here, no model ranking or numerical conclusion is claimed.

The study is limited by PlantVillage's controlled acquisition conditions, possible class and source imbalance, and the mismatch between simulated CPU benchmarking and the diversity of low-end Android hardware. Future work should execute the protocol on representative Android devices, add field-collected imagery and PlantDoc-style external evaluation, study calibration and rejection, investigate distillation and additional compression, expand to more crop species, explore federated learning, and measure the complete camera-to-prediction pipeline.

REFERENCES

[1] D. P. Hughes and M. Salathé, "An open access repository of images on plant health to enable the development of mobile disease diagnostics through machine learning and crowdsourcing," *arXiv preprint arXiv:1511.08060*, 2015.

[2] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," Frontiers in Plant Science, vol. 7, Art. no. 1419, 2016, doi: 10.3389/fpls.2016.01419.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[4] M. Kalyani, M. Manaswini, S. Shevkari, J. Sinkar, L. K. Tyagi, and Y. K. Sharma, "LeafNet: An intelligent system for plant disease

classification with deep learning models," in Proc. 2026 13th Int. Conf. Comput. Sustainable Global Development (INDIACom), New Delhi, India, 2026, pp. 1–6, doi: 10.23919/INDIACom70271.2026.11525518.

[5] M. Kavitha, Y. K. Sharma, S. Shevkari, S. A. Londhe, A. Aggarwal, and H. Garlapati, "Performance analysis of deep learning models—CNN, AlexNet, GAN and Transformer with optimizers for plant disease classification," in Proc. 2026 13th Int. Conf. Comput. Sustainable Global Development (INDIACom), New Delhi, India, 2026, pp. 1–6, doi: 10.23919/INDIACom70271.2026.11525433.

[6] TensorFlow, "Higher accuracy on vision models with EfficientNet-Lite," TensorFlow Blog, 2020.

[7] D. Singh et al., "PlantDoc: A dataset for visual plant disease detection," in Proc. ACM IKDD CoDS and COMAD, 2020, doi: 10.1145/3371158.3371196.

[8] "Advances in deep learning applications for plant disease and pest detection: A review," Remote Sensing, vol. 17, no. 4, Art. no. 698, 2025.

[9] H. Yang et al., "Towards compact ConvNets via structure-sparsity regularized filter pruning," IEEE Trans. Pattern Anal. Mach. Intell., 2020, doi: 10.1109/TPAMI.2019.2913408.

[10] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in Proc. Int. Conf. Learn. Representations (ICLR), 2016.

[11] TensorFlow, "Post-training quantization," TensorFlow Model Optimization Guide, accessed Aug. 25, 2026.

[12] TensorFlow Lite, "Model benchmark tool," Android Open-Source Project Documentation, accessed Aug. 25, 2026.

[13] TensorFlow Datasets, "PlantVillage dataset," documentation, accessed Aug. 25, 2026.

[14] S.-B. Cho et al., "Heterogeneous domain adaptation method for tomato leaf disease classification based on CycleGAN," Journal of Intelligent *& Fuzzy Systems*, 2023, doi: 10.3233/JIFS-230561.