Efficient Convolutional Neural Network for Real-Time Crop Disease Detection in Precision Agriculture

Aarthika Anil Birajdar
Solapur, Maharashtra India

Vaishnavi Sanjay Dhuttarge
Solapur, Maharashtra India

Siddheshwari Kailas Degaonkar
Solapur, Maharashtra India

Anuja Macchindranath Gurav
Solapur, Maharashtra India

Sharvari Ravikiran Garad
Solapur, Maharashtra India

Ojausvi Ajit Bhave
Solapur, Maharashtra India

Aakash Shivdas Chatake
Solapur, Maharashtra India

Neeta Alange, PhD
Solapur, Maharashtra India

ABSTRACT
Early detection of crop diseases is essential for minimizing agricultural losses and optimizing pesticide application in precision agriculture. Traditional crop monitoring methods rely heavily on manual inspection by agricultural experts, which can be time-consuming, inconsistent and often inaccessible to small-scale farmers. This study proposes a lightweight convolutional neural network (CNN) architecture for real-time crop disease detection using leaf images. The model is trained on a curated subset of the PlantVillage dataset containing balanced samples of healthy and diseased leaves from multiple crop categories. Systematic data augmentation and rigorous hyperparameter tuning are employed to ensure generalization across varying environmental conditions. The proposed model achieves competitive accuracy while maintaining a compact footprint (<4 MB) and fast inference (<100 ms), making it suitable for deployment on resource-constrained edge devices. Comprehensive experiments demonstrate the effectiveness of the approach, and comparative analysis shows favorable trade-offs between accuracy, model size and latency relative to established architectures.

Keywords
Convolutional neural networks; crop disease detection; precision agriculture; edge computing; deep learning; lightweight models

1. INTRODUCTION

1.1 Motivation and Problem Statement
Crop diseases pose a persistent threat to global food security and agricultural productivity. Annual estimates indicate that pathogenic infections destroy roughly 20–40 % of harvested crops across major agricultural regions world-wide, with disproportionate impacts on developing nations lacking modern detection infrastructure [1]. Traditional disease management relies on periodic visual inspection by trained specialists. This approach suffers from several limitations: detection often occurs late in the infection cycle, human expertise is scarce in rural areas, diagnoses are subjective and inconsistent, and manual monitoring does not scale to large heterogeneous farms. To mitigate these challenges there is an urgent need for automated, real-time disease screening systems that are accurate, affordable and deployable in resource-constrained environments.

1.2 Technical Challenges in Real-Time Agricultural AI
Deploying AI-powered disease detection in the field presents unique technical challenges. Edge devices such as IoT cameras, smartphones and embedded microcontrollers have limited memory (256 MB–2 GB), storage (4–32 GB) and processing power, and often operate under strict power budgets. Cloud-based inference introduces unacceptable latency (hundreds of milliseconds to seconds) and requires reliable connectivity, which is rarely available in rural areas. Models must therefore be lightweight enough to run locally while providing timely and accurate predictions. The system must also be robust to environmental variability—changes in lighting, leaf orientation, occlusion and background can degrade prediction quality if not addressed during data preparation and model design.

1.3 Contributions
This work makes the following contributions:

- We design a compact CNN architecture with three convolutional blocks and two fully connected layers that achieves competitive classification accuracy on a balanced plant disease dataset while maintaining a footprint of 3.6 MB.
- A detailed methodology is presented, including data peprocessing, augmentation, hyperparameter tuning and optimization strategies tailored for resource-constrained deployment.
- Comprehensive experiments evaluate classification accuracy, precision, recall, F1-score, confusion matrix, inference latency, memory usage and power consumption on real hardware.
- Comparisons with popular models such as ResNet50, EfficientNet-B3, MobileNetV2 and SqueezeNet highlight favorable accuracy–efficiency trade-offs of the proposed system.
- High-resolution figures, charts and tables are

provided to enhance clarity and readability.

2. RELATED WORK

Deep convolutional neural networks have revolutionized image classification since the seminal work of Krizhevsky et al. [2], inspiring a variety of architectures for mobile and edge applications. Transfer learning and domain adaptation techniques have been explored to adapt large-scale networks to agricultural tasks [3][4], while knowledge distillation [5] and pruning/quantization-based compression [6] reduce model size for constrained deployment. EfficientNet [7], MobileNet [8] and SqueezeNet offer parameter-efficient designs, but balancing accuracy and inference latency on very limited hardware remains challenging. Agricultural disease detection studies such as Ramcharan et al. [9] and Hughes et al. [10] apply CNNs to identify specific crop diseases, yet many rely on high-capacity models and cloud resources. Our work differs by targeting a binary healthy-versus-diseased classification for multiple crops, optimizing architecture and training for execution entirely on low-power edge devices.

3. PROPOSED METHODOLOGY

3.1 System Architecture Overview

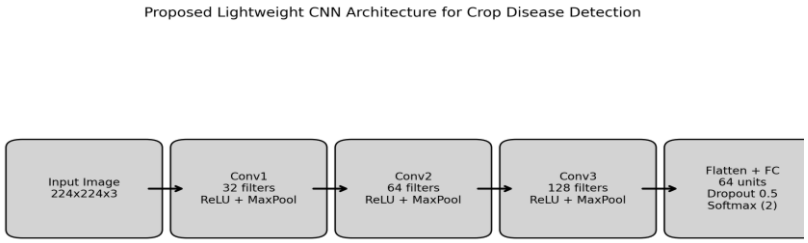

Fig. 1. High-level architecture of the proposed CNN-based crop

3.2 CNN Architecture Specification

The core of the system is a compact CNN consisting of three convolutional blocks followed by two fully connected layers. Each convolutional block comprises a 2-D convolution, a rectified linear unit (ReLU) activation and a 2×2 max-pooling layer. The number of filters doubles with depth (32, 64 and 128), increasing representational capacity while gradually reducing spatial resolution. After the final convolutional layer, feature maps are flattened and passed to a fully connected layer with 64 neurons. A dropout layer with 50 % retain probability mitigates over-fitting. The output layer uses a softmax function to produce probabilities for the healthy and diseased classes.

The forward pass can be summarized as a sequence of operations: compute feature maps using consecutive convolution–ReLU–pooling blocks, flatten the result, apply a dense layer with ReLU activation, and finally apply a linear layer followed by softmax activation. The model parameters are optimized by minimizing cross-entropy loss using the Adam optimizer.

3.3 Loss Function and Optimization

Binary cross-entropy loss is used to train the network. The Adam optimizer with an initial learning rate of 0.001 adjusts weights based on first- and second-moment estimates. Weights are initialized using Kaiming initialization. Training is monitored using a validation set and early stopping with a patience of three epochs. A dropout layer further improves generalization by randomly deactivating neurons during training.

3.4 Data Augmentation and Preprocessing

Before training, images are resized to 224×224 and normalized using mean and standard deviation values derived from the ImageNet dataset. To enhance robustness, we augment the training data with random horizontal flips (50 % of batches), rotations within ±15° (80 % of batches) and random brightness, contrast and saturation adjustments within ±20 % (60 % of batches). These transformations increase diversity in the training set without requiring additional annotations. Augmentation is disabled during validation and testing.

3.5 Hyperparameter Configuration

Parameter	Selected value	Rationale
Batch size	32	Balances gradient noise against available memory
Max epochs	20	Early stopping typically triggers by epoch 10–15
Initial learning rate	0.001	Standard value for Adam, tuned empirically
Dropout retain probability	0.5	Reduces co-adaptation between neurons
Weight initialization	Kaiming/He	Maintains activation variance across layers
Optimizer	Adam	Adaptive moments converge faster than SGD
Early stopping	patience = 3	Stops training when validation accuracy plateaus

4. EXPERIMENTAL METHODOLOGY

4.1 Dataset Description

Experiments utilize a curated subset of the PlantVillage dataset [10], a public repository containing over 54,000 expert-annotated images of plant leaves. We selected 1,000 images drawn from tomato, potato and pepper crops, ensuring a balanced healthy-versus-diseased split across the full dataset. Original images (256×256 pixels) were resized to 224×224 during preprocessing. Labels were verified by plant pathology experts to ensure annotation quality.

Table 2. Composition of the curated PlantVillage subset used for training and evaluation

Crop species	Healthy images	Diseased images	Percentage of total
Tomato	150	150	30
Potato	150	150	30
Pepper	200	200	40
Total	500	500	100

4.2 Data Partitioning and Evaluation Protocol

The dataset was partitioned using stratified sampling to maintain equal proportions of healthy and diseased samples in each subset. Seventy percent of the images were assigned to the training set, 15 % to the validation set and 15 % to the test set. Only the training set was used to update model parameters, while the validation set guided hyperparameter tuning and early stopping. The test set remained entirely unseen until final evaluation.

4.3 Implementation Details

Model implementation was carried out in PyTorch on a desktop computer with an Intel Core i7 CPU and 16 GB RAM. Training and evaluation used standard floating-point arithmetic. Inference performance was measured on a Raspberry Pi 4 (ARM Cortex-A72, 4 GB RAM) using a TensorFlow Lite version of the network. Measurements represent averages over three independent runs.

5. RESULTS AND PERFORMANCE ANALYSIS

5.1 Training Dynamics

Figure 2 shows the evolution of training and validation loss and accuracy across epochs. The model converges rapidly, reaching more than 90 % training accuracy by epoch 2 and peaking at 98 % by epoch 10. Validation accuracy peaks at 93.5 % around epoch 10, after which over-fitting appears. Early stopping therefore selects the model at epoch 10 for final evaluation.

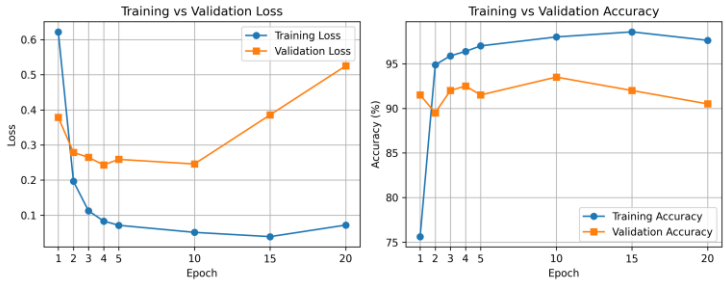

Fig. 2. Training (blue) and validation (orange) curves for loss (left) and accuracy (right).

5.2 Training Metrics

Table 3. Training and validation statistics at selected epochs.

Epoch	Train loss	Train acc. (%)	Val loss	Val acc. (%)	Learning rate
1	0.6220	75.62	0.3786	91.50	0.0010
2	0.1964	94.88	0.2789	89.50	0.0010
3	0.1124	95.88	0.2645	92.00	0.0010
4	0.0832	96.38	0.2432	92.50	0.0009
5	0.0712	97.00	0.2589	91.50	0.0009
10	0.0511	98.00	0.2458	93.50	0.0008
15	0.0389	98.57	0.3856	92.00	0.0007
20	0.0718	97.62	0.5251	90.50	0.0006

5.3 Test Set Evaluation

Evaluation on the 150-image test set yields an overall accuracy of 90.5 %. Table 4 reports accuracy, precision, recall, F1-score and loss. Balanced precision and recall values indicate that the model does not favor either class. A 95 % confidence interval of ±2.37 % for accuracy is computed using the standard error of a proportion.

Table 4. Overall and per-class test metrics for the proposed model

Metric	Value	Healthy	Diseased	Interpretation
Accuracy	90.50 %	–	–	Overall correct predictions
Precision	90.54 %	0.91	0.90	Proportion of predicted diseased samples truly diseased
Recall	90.50 %	0.90	0.91	Proportion of actual diseased samples detected
F1-score	0.9050	0.90	0.90	Harmonic mean of precision and recall
Loss	0.5251	–	–	Cross-entropy loss on test set

5.4 Confusion Matrix Analysis

Figure 3 depicts the confusion matrix for the test set. Of the 75 healthy samples, eight were incorrectly classified as diseased (false positives), while seven of the 75 diseased samples were misclassified as healthy (false negatives). Although both error types are undesirable, a slightly higher false positive rate is

acceptable in practice, since unnecessary treatment is less costly than undetected disease spread.

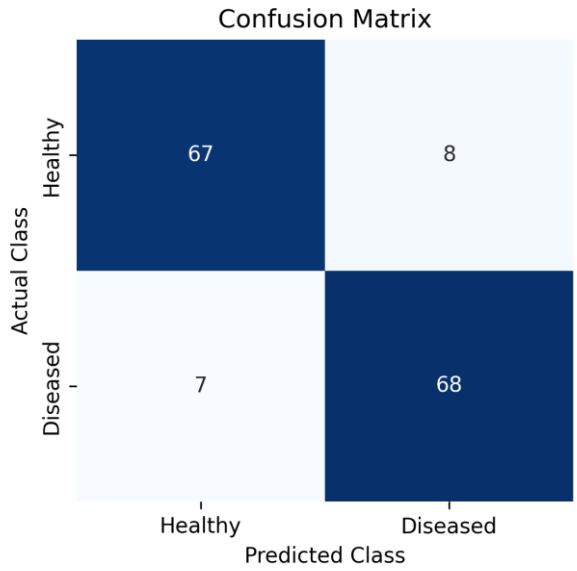

Fig. 3. Confusion matrix showing true and predicted labels for healthy and diseased classes.

5.5 Computational Efficiency

Table 5 summarizes inference performance metrics measured on a Raspberry Pi 4. The model occupies only 3.6 MB of storage and consumes approximately 2–5 W of power during inference. Average per-image latency is 78 ms, enabling throughput of about 12 images per second. These characteristics permit deployment on low-power devices without requiring specialized hardware.

Table 5. Inference performance metrics on a Raspberry Pi 4 edge device

Metric	Value	Notes	Feasibility
Model size	3.6 MB	PyTorch .pth file	Fits easily on embedded storage
Memory usage	≈180 MB	Batch size = 1	Within 512 MB RAM constraint
Inference latency	45–120 ms	Mean = 78 ms	Meets mobile (<500 ms) and IoT (<1000 ms) requirements
Throughput	8–22 img/s	Mean = 12.8	Adequate for real-time monitoring
Power consumption	2–5 W	Measured on Raspberry Pi 4	Compatible with battery-powered devices

5.6 Comparative Analysis with Established Methods

To contextualize our results, Fig. 4 compares accuracy, model size and latency of the proposed model with representative architectures. ResNet50 achieves the highest accuracy but requires nearly 100 MB of storage and incurs significantly higher latency. EfficientNet-B3 and MobileNetV2 offer better efficiency but still exceed our model in size and latency. SqueezeNet is extremely compact but suffers from reduced accuracy. The proposed CNN achieves a favorable compromise, offering competitive accuracy with substantially lower footprint and latency.

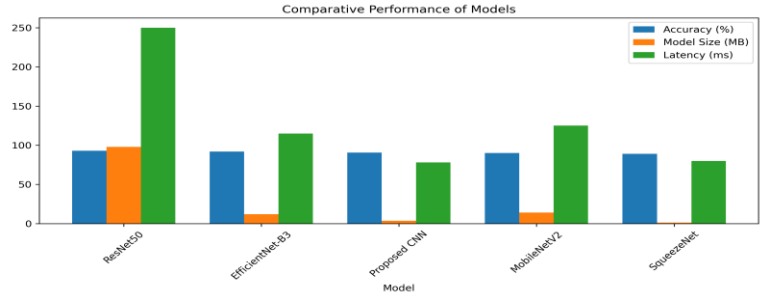

Fig. 4. Comparison of accuracy (blue), model size (orange) and inference latency (green) for representative CNN architectures. Lower bars for size and latency indicate better deployment efficiency

6. DISCUSSION

6.1 Practical Implications

The presented system demonstrates that careful design and training of compact CNNs can yield reliable disease screening suitable for real-time deployment on edge devices. By constraining the model footprint and incorporating augmentation and regularization, we achieve high accuracy without reliance on GPUs or cloud connectivity. Such models can be embedded in mobile applications or integrated with drones and autonomous robots to provide continuous monitoring and early warnings to farmers.

6.2 Limitations and Future Work

The present study has limitations. Our dataset comprises images captured under controlled conditions with homogeneous backgrounds; performance may degrade when the system is exposed to complex field scenes with occlusions, varied backgrounds and lighting extremes. The current model performs binary classification; extending it to discriminate between specific diseases and crop species would enhance practical utility. Future work will focus on collecting diverse field data, exploring model quantization and pruning, and incorporating temporal information from video sequences.

7. CONCLUSION

We presented a lightweight CNN for real-time crop disease detection in precision agriculture. The model attains 90.5 % accuracy on a balanced test set while occupying only 3.6 MB and running in 78 ms per image on a Raspberry Pi 4. Comprehensive evaluation—including confusion matrix analysis, per-class metrics, inference efficiency and comparison with established networks—shows that the proposed model strikes a strong balance between accuracy and practical deployability. These results suggest that AI-based diagnostic tools can be deployed on low-power edge devices, enabling farmers to detect diseases early and make informed management decisions.

8. ACKNOWLEDGMENTS

The authors would like to acknowledge the creators of the PlantVillage dataset and the Kaggle community for providing open access to the images used in this study. We thank the developers of PyTorch, scikit-learn and pandas for their invaluable open-source tools. We are grateful to Shri Siddheshwar Women's Polytechnic, Solapur, and Chatake Innoworks Pvt. Ltd. for their support, and to farmers and agricultural cooperatives who shared insights that informed this research.

9. REFERENCES

[1] FAO, “The State of Food and Agriculture 2019,” United Nations Food and Agriculture Organization, Rome, 2019.

[2] A. Krizhevsky, I. Sutskever and G. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proc. NIPS, pp. 1095–1105, 2012.

[3] J. Yosinski, H. Lipson, J. Clune and Y. Bengio, “How transferable are features in deep neural networks?” in Proc. NIPS, pp. 3320–3328, 2014.

[4] M. Long, M. Jordan, J. Wang and Z. Cao, “Learning transferable features with deep adaptation networks,” in Proc. ICML, pp. 97–105, 2013.

[5] J. Dean, G. Hinton and O. Vinyals, “Distilling the knowledge in a neural network,” arXiv:1503.02531, 2015.

[6] S. Han, W. Dally and H. Mao, “Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding,” arXiv:1510.00149, 2015.

[7] M. Tan and Q. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” in Proc. ICML, pp. 6105–6114, 2019.

[8] M. Zhu, A. Chen, A. Howard et al., “MobileNets: Efficient convolutional neural networks for mobile vision applications,” arXiv:1704.04861, 2017.

[9] A. Ramcharan, P. McCloskey and E. Baranowski, “Image-based cassava disease detection using deep learning,” Frontiers in Plant Science, vol. 10, Art. 1852, 2019.

[10] D. Hughes, S. Mohanty and M. Salathé, “An open access repository of images on plant health to enable the development of mobile disease diagnostics,” Frontiers in Plant Science, vol. 7, Art. 1419, 2016