WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Cotton Leaf Disease Detection Using Machine Learning and Raspberry Pi

Deepak R Khadse, Shubham S Pawar

[a] Department of Electronics and Telecommunication Engineering G H Raisoni College of Engineering and Management Jalgaon
Gate No. 57 / Gat No. 57/1, Shirsoli Road, Tambapura, Mohadi, Jalgaon – 425002 / 425003, Maharashtra, India
[b] Department of Electronics and Telecommunication Engineering G H Raisoni College of Engineering and Management Jalgaon
Gate No. 57 / Gat No. 57/1, Shirsoli Road, Tambapura, Mohadi, Jalgaon – 425002 / 425003, Maharashtra, India

A B S T R A C T :

Biotic stresses induced by foliar pathogens and phytophagous insects severely constrain the lint yield and fiber quality of upland cotton (Gossypium hirsutum L.) across global agricultural belts. Conventional field diagnosis relies on manual visual scouting by extension agronomists, a procedure that is time-intensive, prone to inter-observer subjectivity, and largely unavailable to smallholder cultivators in remote regions. While deep learning classifiers have demonstrated high diagnostic accuracy in laboratory settings, existing cloud-dependent architectures require high-bandwidth internet connectivity and graphics processing unit (GPU) servers that are impractical for offline field deployment. This paper presents the design, mathematical formulation, and hardware implementation of a standalone edge-computing diagnostic system for automated cotton leaf disease and pest classification using a custom five-block Convolutional Neural Network (CNN) deployed on a Raspberry Pi 4 Model B (4 GB RAM) interfaced with a 3.5-inch resistive touch display. A balanced dataset of 3,600 RGB foliar images spanning six categories—Target Spot, Powdery Mildew, Healthy foliage, Bacterial Blight, Armyworm damage, and Aphid infestation—was curated and partitioned into 2,880 training, 360 validation, and 360 testing samples. Input frames were standardized to 224 × 224 × 3 resolution via OpenCV and NumPy pipelines incorporating bilateral filtering, floating-point normalization, and stochastic affine augmentation. Trained over 30 epochs (batch size 32, learning rate 0.001, Adam optimizer, categorical cross-entropy loss), the network achieved 96.8% training accuracy, 94.6% validation accuracy, and 93.9% independent testing accuracy, with macro-averaged precision, recall, and F1-scores of 94.2%, 94.0%, and 94.2%, respectively. Serialized into an 18.6 MB footprint (4.85 million parameters), the model required 2.84 seconds for one-time boot initialization on the Raspberry Pi 4 and executed image preprocessing in 38 milliseconds and forward-pass inference in 412 milliseconds, delivering a total per-leaf prediction time of 0.45 seconds entirely offline.

Keywords: Cotton Leaf Disease Detection, Convolutional Neural Network (CNN), Deep Learning, Edge AI, Raspberry Pi 4, Computer Vision, OpenCV, Precision Agriculture.

1. Introduction

Upland cotton (Gossypium hirsutum L.) serves as the primary natural fiber source for the global textile manufacturing sector and underpins the agrarian economy of major producing nations, including India, China, the United States, and Pakistan [1, 2]. Throughout its vegetative expansion and boll-maturation stages, the cotton canopy is continuously exposed to diverse biotic stresses. Foliar infections caused by bacterial and fungal pathogens, combined with sap-sucking and defoliating insect pests, reduce global cotton harvests by 15% to 30% annually [3, 4]. Unchecked foliar lesions impair stomatal conductance, degrade palisade chlorophyll complexes, trigger premature abscission of reproductive squares and bolls, and diminish the tensile strength and micronaire index of harvested lint.

Five biotic conditions account for a major share of foliar degradation in cotton cultivation: Bacterial Blight (Xanthomonas citri pv. malvacearum), Target Spot (Corynespora cassiicola), Powdery Mildew (Leveillula taurica), Aphids (Aphis gossypii), and Armyworms (Spodoptera frugiperda) [1, 3]. Conventional crop protection workflows rely on periodic visual scouting by farmers or regional agricultural extension officers. However, ocular diagnosis exhibits three critical bottlenecks. First, early necrotic symptoms share overlapping morphological traits; coalescing angular lesions of Bacterial Blight closely resemble the circular necrotic blotches of Target Spot [4]. Second, certified plant pathologists are scarce in rural farming clusters, delaying intervention during acute epidemic windows. Third, laboratory assays such as polymerase chain reaction (PCR) amplification and enzyme-linked immunosorbent assays (ELISA) require specialized reagents and multi-day turnaround times [5]. Without rapid on-site diagnostics, cultivators frequently resort to broadcast applications of broad-spectrum fungicides and organophosphate insecticides, escalating production costs and accelerating agrochemical runoff.

Deep representation learning and computer vision have enabled automated plant disease classification directly from RGB leaf imagery [6, 7]. Convolutional Neural Networks (CNNs) replace fragile handcrafted feature extractors by learning hierarchical spatial kernels that capture venation discoloration, chlorotic halos, and necrotic margins [8, 10]. Nevertheless, most high-performing models in recent literature rely on computationally

intensive backbones—such as VGG-19, ResNet-152, compound-scaled EfficientNets, or Vision Transformers (ViTs)—that demand cloud GPU servers and stable broadband connectivity [3, 5]. Because remote agricultural belts often lack reliable cellular data coverage, cloud-dependent mobile applications fail during real-world field scouting [19, 20].

1.1. Statement of technical novelty and contributions

Unlike prior studies that evaluate heavyweight transfer-learning ensembles exclusively on workstation GPUs [1, 4] or restrict edge classification to three or four pathogen-only categories [14, 19], this study co-designs a receptive-field-matched CNN topology and an embedded runtime service tailored for the quad-core ARM Cortex-A72 processor of the Raspberry Pi 4 Model B (4 GB RAM). The primary technical contributions of this paper are:

- **Unified Pathogen-and-Pest Dataset Curation:** Compilation and stratified 80:10:10 partitioning of a balanced 3,600-image cotton leaf dataset (224 × 224 × 3 resolution) spanning two fungal diseases (Target Spot, Powdery Mildew), one bacterial infection (Bacterial Blight), two insect pests (Armyworm, Aphids), and Healthy foliage.
- **Receptive-Field-Aligned 18.6 MB Custom CNN:** Formulation of a five-block convolutional topology (4.85 million parameters, serialized to 18.6 MB in IEEE-754 float32 HDF5 format) whose terminal receptive field (94 × 94 pixels) captures both micro-lesion borders and macro-lamina deformations, achieving 93.9% test accuracy and a 94.2% macro F1-score.
- **Persistent Singleton Edge Daemon on Raspberry Pi 4:** Hardware-software integration on a Raspberry Pi 4 (4 GB) with a 3.5-inch SPI resistive touch LCD, isolating the 2.84-second cold-boot model load from the active scan loop to achieve 38 ms OpenCV preprocessing and 412 ms CPU inference (0.45 s total prediction time) entirely offline.

2. Literature review

Early automated plant disease recognition pipelines relied on deterministic lesion segmentation—such as Otsu dynamic thresholding, K-means clustering, and watershed morphology—paired with handcrafted Gray-Level Co-occurrence Matrix (GLCM) texture descriptors, Local Binary Patterns (LBP), and color histograms classified via Support Vector Machines (SVM) [5]. While computationally inexpensive, handcrafted features degraded sharply under variable ambient lighting, specular leaf glare, and complex soil backgrounds.

Deep convolutional networks overcame manual feature engineering by learning end-to-end spatial hierarchies. Zekiwos and Bruck [6] developed a deep CNN for cotton leaf disease and pest diagnosis, attaining 96.4% accuracy across four classes. Latif et al. [9] combined deep learning feature extraction with Genetic Algorithm (GA) selection for cotton disease recognition, while Kinger and Tagalpallewar [7] validated sequential CNNs on foliar lesions (93.2% accuracy). Rai and Pahuja [8] proposed an improved deep CNN that achieved 97.4% accuracy on diseased cotton leaves and plants. Otiya et al. [11] and Vidivelli et al. [12] integrated Gradient-weighted Class Activation Mapping (Grad-CAM) with deep CNNs to confirm that spatial activations localized on necrotic leaf tissue rather than background artifacts.

Between 2024 and 2026, investigations focused heavily on transfer learning, metaheuristic optimization, and Vision Transformer hybrids. Anwar et al. [13] benchmarked six pre-trained networks (DenseNet121, DenseNet169, MobileNetV2, ResNet50V2, VGG16, and VGG19) on cotton foliage across resolutions from 32 × 32 to 256 × 256, showing that DenseNet169 reached 96% accuracy at 256 × 256 while MobileNetV2 dropped to 52% at 32 × 32. Johri et al. [1] evaluated transfer learning backbones on the six-class Kaggle Cotton Plant Disease (CPD) dataset (Aphids, Armyworm, Bacterial Blight, Powdery Mildew, Target Spot, and Healthy), pairing contour-based region-of-interest (ROI) extraction with EfficientNetB3 to report 99.96% training accuracy. Vangar et al. [10] further demonstrated transfer-learning-based cotton disease classification for precision agriculture. In July 2025, Rehman et al. [2] integrated Temporal Logic of Actions (TLA+) formal verification with a five-layer CNN (256 × 256 input) across the same six cotton classes, achieving 98.7% accuracy while noting lower class-specific F1-scores on Armyworm (0.87) and Powdery Mildew (0.90). Hussen et al. [4] combined EfficientNetB3 and InceptionResNetV2 with GA hyperparameter tuning and SMOTE balancing on 2,637 cotton images to attain 98.0% accuracy alongside LIME and SHAP explainability. Dhruw et al. [3] fused a six-block CNN with Vision Transformer self-attention across eight cotton classes (98.5% accuracy), and Xu et al. [5] introduced ViTKAB, integrating Kolmogorov-Arnold Networks (KAN) with BiFormer sparse dynamic attention (98.05% accuracy). In 2026, Reddy and Kumar [14] combined Kalman filtering, Fuzzy Local Information C-Means (FLICM) segmentation, and an EfficientNet–Fuzzy Shepard CNN (EffNet_ShCNN) to classify three cotton diseases with 91.70% accuracy, while Nazim et al. [15] evaluated an EfficientNet + Grad-CAM/LIME pipeline on 800 cotton leaf images (224 × 224), reporting 92.0% validation accuracy and 88.0% test accuracy.

In parallel, researchers investigated embedded edge deployment on single-board computers. Dodda et al. [16] built a Raspberry Pi CNN prototype for automated plant disease detection (92.5% accuracy). Saad and Aidi [17] and Tanksale and Mane [18] benchmarked CNN architectures (Basic CNN, AlexNet, GoogLeNet, VGGNet, ResNet-50, and MobileNet) on Raspberry Pi 4 and Raspberry Pi 5 RISC platforms, finding that over-parameterized networks incur multi-second latencies on ARM CPUs. Conversely, Hammam et al. [19] validated an embedded CNN for potato leaf classification on Raspberry Pi hardware (96.1% accuracy), and Bari et al. [20] demonstrated an offline Raspberry Pi touchscreen detector using MobileNetV2 across 14 crop classes with sub-two-second latency. Table 1 summarizes these recent studies.

Table 1 - Summary of recent studies on cotton and plant leaf disease detection (2021–2026).

Author(s) & Year	Core Methodology	Classes	Accuracy	Identified Research Gap / Edge Limitation
Zekiwos & Bruck (2021) [6]	Custom Deep CNN + Image Processing	4	96.4%	Evaluated on desktop PC; no standalone embedded hardware deployment.
Kinger & Tagalpallewar (2022) [7]	Deep Convolutional Neural Network	4	93.2%	Limited pest class coverage; not benchmarked for edge latency or memory size.
Rai & Pahuja (2023) [8]	Improved Deep CNN Architecture	4	97.4%	Workstation GPU execution; lacked embedded ARM CPU latency verification.
Dodda et al. (2024) [16]	CNN on Raspberry Pi Platform	5	92.5%	General crop focus; lacked millisecond stage latency profiling on cotton pests.
Johri et al. (2024) [1]	EfficientNetB3 + Contour ROI Extraction	6	99.96% (Train)	Heavyweight transfer learning backbone with multi-stage contour segmentation.
Rehman et al. (2025) [2]	5-Layer CNN (256 × 256) + TLA+ Model	6	98.7%	Lower F1 on Armyworm (0.87) and Powdery Mildew (0.90); no edge hardware testbed.
Hussen et al. (2025) [4]	EfficientNetB3 + InceptionResNetV2 + GA	6	98.0%	Dual-backbone ensemble increases memory overhead for single-board computers.
Dhruw et al. (2025) [3]	Hybrid CNN-ViT Architecture	8	98.5% (Hybrid)	Transformer attention blocks introduce high compute cost on non-GPU edge nodes.
Hammam et al. (2025) [19]	Embedded CNN on Raspberry Pi	3	96.1%	Validated embedded CNN on potato leaves; did not cover multi-class cotton pests.
Reddy & Kumar (2026) [14]	EfficientNet–Fuzzy ShCNN + FLICM	3	91.7%	Restricted to 3 classes; iterative fuzzy clustering adds preprocessing overhead.
Nazim et al. (2026) [15]	EfficientNet (224 × 224) + Grad-CAM	4	88.0% (Test)	Small dataset (800 images) reduced independent test generalization (88.0%).

2.1. Synthesis of identified research gaps

Synthesizing the surveyed literature reveals three core research gaps addressed in this work:

- **Cloud-Edge Compute Asymmetry:** High-accuracy dual-backbone ensembles [4] and CNN-ViT hybrids [3, 5] depend on server-grade GPUs, whereas existing Raspberry Pi deployments either omit cotton-specific pest classes or exhibit inference latencies of 1.5–2.0 seconds [16, 20].
- **Fragmented Pathogen vs. Pest Taxonomies:** Several cotton disease studies classify only three or four fungal/bacterial conditions [5, 14], neglecting co-occurring insect pests (Armyworm and Aphids) that require distinct insecticide regimens.
- **Lack of Stage-Decoupled Edge Latency Profiling:** Prior embedded studies rarely separate one-time model loading latency from per-frame OpenCV preprocessing and ARM CPU inference times.

3. Proposed methodology

The proposed cotton leaf diagnostic framework executes an eight-stage workflow transitioning from offline model training to standalone edge deployment on a Raspberry Pi 4 (4 GB): Dataset Collection → Image Preprocessing → CNN Training → Model Evaluation → Model Saving → Raspberry Pi Deployment → Image Classification → Prediction Display.

3.1. Stage-wise operational pipeline

The operational sequence was implemented across eight deterministic stages:

- **1. Dataset Collection:** A balanced repository of 3,600 RGB cotton leaf images was compiled across six categories: Target Spot, Powdery Mildew, Healthy, Bacterial Blight, Armyworm, and Aphids (600 images per class).
- **2. Image Preprocessing:** Using OpenCV and NumPy, raw frames were converted from BGR to RGB, resized via bicubic interpolation to 224 × 224 × 3, denoised via bilateral filtering, and normalized to [0, 1]. Stochastic affine augmentation was applied to the training split.
- **3. CNN Training:** The 2,880 training and 360 validation tensors were used to train the five-block CNN in TensorFlow/Keras over 30 epochs (batch size 32, Adam optimizer at $\eta = 0.001$, categorical cross-entropy loss).
- **4. Model Evaluation:** The converged network was tested on 360 unseen hold-out images (60 per class) using overall accuracy, precision, recall, F1-score, and a 6 × 6 confusion matrix.
- **5. Model Saving:** Stripping optimizer states, the trained graph and weights were serialized into an 18.6 MB HDF5 artifact (cotton_cnn_model.h5).
- **6. Raspberry Pi Deployment:** The 18.6 MB model was transferred to a Raspberry Pi 4 Model B (4 GB RAM) running 64-bit Raspberry Pi OS and loaded into RAM via a persistent boot daemon.

- **7. Image Classification:** Each captured leaf frame underwent OpenCV preprocessing in 38 ms and multi-threaded Cortex-A72 forward propagation in 412 ms, completing prediction in 0.45 seconds.
- **8. Prediction Display:** The predicted class label and confidence score were rendered onto the 3.5-inch resistive touch LCD.

3.2. Mathematical formulation of preprocessing and augmentation

Let an acquired raw RGB foliar frame be denoted as $I_{raw} \in \{0, ..., 255\}^{(H0 \times W0 \times 3)}$. Bicubic interpolation over a 4 × 4 pixel neighborhood maps I_{raw} onto the standardized spatial lattice $I_{resized} \in R^{(224 \times 224 \times 3)}$. An edge-preserving bilateral filter then attenuates high-frequency sensor noise while preserving sharp lesion boundaries:

$$I_{filt}(p) = (1 / W_p) \sum_{q \in \Omega} I_{resized}(q) \exp(-||p - q||^2 / 2\sigma_s^2) \exp(-||I_{resized}(p) - I_{resized}(q)||^2 / 2\sigma_r^2) \quad (1)$$

where $p, q \in \Omega$ are pixel coordinates in a 5 × 5 neighborhood, $\sigma_s = 15$, $\sigma_r = 75$, and W_p is the normalization factor. Next, vectorized NumPy scaling maps intensities to 32-bit floating-point tensors in [0, 1]:

$$X(y, x, c) = float32(I_{filt}(y, x, c)) / 255.0, \quad X \in [0, 1]^{(224 \times 224 \times 3)} \quad (2)$$

During training, stochastic affine transformations—rotations $\theta \sim U(-20°, +20°)$, isotropic scaling $s \sim U(0.90, 1.10)$, translations $t_x, t_y \sim U(-22, +22)$ pixels (±10%), and horizontal/vertical flips ($p = 0.50$)—were applied dynamically:

$$[x', y', 1]^T = [\,[s\cos\theta, -s\sin\theta, t_x], [s\sin\theta, s\cos\theta, t_y], [0, 0, 1]\,] \cdot [x, y, 1]^T \quad (3)$$

4.Dataset description

To prevent majority-class bias [1, 4], the dataset was curated with exactly 600 RGB images per category (3,600 total images) following the six-class Cotton Plant Disease taxonomy [1, 2]. Stratified random partitioning allocated 2,880 images (80%) to training, 360 images (10%) to validation, and 360 images (10%) to independent testing (Table 2).

Table 2 - Stratified dataset partitioning across six cotton leaf categories (224 × 224 × 3).

Class ID	Category Name	Causal Agent / Etiology	Train (80%)	Val (10%)	Test (10%)	Total Images
0	Target Spot	Corynespora cassiicola (Fungal)	480	60	60	600
1	Powdery Mildew	Leveillula taurica (Fungal)	480	60	60	600
2	Healthy	Uninfected Control Foliage	480	60	60	600
3	Bacterial Blight	Xanthomonas citri (Bacterial)	480	60	60	600
4	Armyworm	Spodoptera frugiperda (Pest)	480	60	60	600
5	Aphids	Aphis gossypii (Pest)	480	60	60	600
All	**Six Cotton Classes**	**Balanced Stratified Split**	**2,880**	**360**	**360**	**3,600**

4.1. Pathological and morphological symptomatology

Each category in the curated dataset manifests distinct visual features captured during representation learning:

- **Target Spot (Corynespora cassiicola):** Circular brown necrotic lesions (2–10 mm) displaying concentric dark target rings and yellow chlorotic halos [1, 4].
- **Powdery Mildew (Leveillula taurica):** White powdery fungal mycelia and conidia coating adaxial and abaxial leaf surfaces, restricting photosynthesis [1].
- **Healthy Foliage:** Uniform green chlorophyll pigmentation, intact palmate venation, and unblemished margins.
- **Bacterial Blight (Xanthomonas citri pv. malvacearum):** Angular, vein-bounded water-soaked spots that turn dark brown or black and coalesce along major veins [1, 3].
- **Armyworm (Spodoptera frugiperda):** Irregular chewed perforations and severe interveinal leaf skeletonization caused by larval feeding [4].
- **Aphids (Aphis gossypii):** Downward leaf cupping, crinkling, chlorotic stippling, and honeydew/sooty mold deposits caused by sap extraction [2].

5. CNN architecture

To achieve high multi-class discrimination within a sub-500 ms execution budget on the Raspberry Pi 4 Cortex-A72 CPU, a custom five-block sequential CNN was architected in TensorFlow/Keras. Unlike VGG-16 (138 million parameters, 528 MB), the proposed network contains 4,850,026 parameters. Stored in 32-bit IEEE-754 floating-point precision (4 bytes/parameter = 18.50 MB) plus 0.10 MB of HDF5 structural metadata, the serialized model occupies exactly 18.6 MB.

5.1. Layer topology and receptive field progression

The network processes 224 × 224 × 3 inputs through five convolutional blocks (Table 3). Each block applies a 3 × 3 Conv2D layer (same padding), Batch Normalization, ReLU activation, and 2 × 2 MaxPooling2D (stride 2), expanding channel depth (32 → 64 → 128 → 128 → 256) while contracting spatial dimensions from 224 × 224 to 7 × 7. The effective receptive field R_l = R_{l-1} + (k_l - 1) ∏_{i=1}^{l-1} s_i expands from 3 × 3 in Block 1 to 22 × 22 in Block 3 (capturing concentric Target Spot rings and angular Bacterial Blight borders) and reaches 94 × 94 at Pool 5 (capturing macroscopic Aphid cupping and Armyworm skeletonization). The flattened 12,544-element vector passes through two regularized dense blocks (340 and 128 units, Dropout 0.50 and 0.30) before the 6-unit Softmax output layer.

Table 3 - Layer-wise architecture, receptive field, and parameter count of the proposed CNN.

Layer / Block	Operation & Kernel Specification	Output Shape	Activation	Parameters
Input Layer	Standardized RGB Image Tensor	224 × 224 × 3	—	0
Conv Block 1	Conv2D (32 × 3 × 3, pad=same) + BatchNorm	224 × 224 × 32	ReLU	1,024
Pool 1	MaxPooling2D (2 × 2, stride=2) [R = 4 × 4]	112 × 112 × 32	—	0
Conv Block 2	Conv2D (64 × 3 × 3, pad=same) + BatchNorm	112 × 112 × 64	ReLU	18,752
Pool 2	MaxPooling2D (2 × 2, stride=2) [R = 10 × 10]	56 × 56 × 64	—	0
Conv Block 3	Conv2D (128 × 3 × 3, pad=same) + BatchNorm	56 × 56 × 128	ReLU	74,368
Pool 3	MaxPooling2D (2 × 2, stride=2) [R = 22 × 22]	28 × 28 × 128	—	0
Conv Block 4	Conv2D (128 × 3 × 3, pad=same) + BatchNorm	28 × 28 × 128	ReLU	148,096
Pool 4	MaxPooling2D (2 × 2, stride=2) [R = 46 × 46]	14 × 14 × 128	—	0
Conv Block 5	Conv2D (256 × 3 × 3, pad=same) + BatchNorm	14 × 14 × 256	ReLU	296,192
Pool 5	MaxPooling2D (2 × 2, stride=2) [R = 94 × 94]	7 × 7 × 256	—	0
Flatten Layer	Spatial Feature Map Flattening	12,544	—	0
Dense Block 1	Fully Connected (340 units) + BatchNorm	340	ReLU	4,266,660
Dropout 1	Stochastic Regularization (p = 0.50)	340	—	0
Dense Block 2	Fully Connected (128 units) + BatchNorm	128	ReLU	44,160
Dropout 2	Stochastic Regularization (p = 0.30)	128	—	0
Output Layer	Fully Connected Classifier (6 units)	6	Softmax	774
Total / Footprint	**Trainable + BatchNorm Parameters (18.6 MB HDF5)**	**6 Classes**	**Softmax**	**4,850,026**

5.2. Mathematical principles of forward propagation and optimization

For convolutional layer l, feature map Z_k^(l) is computed by 2D discrete cross-correlation, batch-normalized over mini-batch B (M = 32), and activated via ReLU:

$$Z_k^{(l)}(i,j) = \sum_{c=1}^{C_{l-1}} \sum_{u=-1}^{1} \sum_{v=-1}^{1} W_{k,c}^{(l)}(u,v)\, A_c^{(l-1)}(i+u, j+v) + b_k^{(l)} \quad (4)$$

$$A_k^{(l)}(i,j) = \max(0, \gamma_k \cdot (Z_k^{(l)}(i,j) - \mu_B) / \sqrt{\sigma_B^2 + \varepsilon}) + \beta_k \quad (5)$$

The output Softmax layer computes posterior probabilities y_hat_c, and network weights Θ are optimized via Adam (η = 0.001, β_1 = 0.9, β_2 = 0.999) to minimize Categorical Cross-Entropy loss L_CCE:

$$\hat{y}_c = \exp(z_c) / \sum_{j=1}^{6} \exp(z_j), \quad L_{CCE}(\Theta) = -(1/M) \sum_{m=1}^{M} \sum_{c=1}^{6} y_{m,c} \log(\hat{y}_{m,c}) \quad (6)$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) \nabla_\Theta L_t, \quad v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla_\Theta L_t)^2, \quad \Theta_{t+1} = \Theta_t - \eta \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \varepsilon) \quad (7)$$

Table 4 lists the configured CNN training hyperparameters.

Table 4 - CNN training hyperparameters and optimization configuration.

Hyperparameter / Specification	Configured Value / Method
Input Image Dimensions	224 × 224 × 3 (RGB Color Space, Normalized [0, 1])
Batch Size (M)	32 Images per Iteration (90 Iterations per Training Epoch)
Total Training Epochs	30 Epochs
Initial Learning Rate (η)	0.001 (1×10^{-3})
Optimizer	Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-7}$)
Loss Function	Categorical Cross-Entropy (L_CCE)
Regularization Mechanisms	Batch Normalization + Affine Augmentation + Dropout (0.50, 0.30)

Hyperparameter / Specification	Configured Value / Method
Serialized Model Footprint	18.6 MB (float32 HDF5 Format)

6. System design

The portable diagnostic unit integrates a Raspberry Pi 4 Model B (4 GB LPDDR4-3200 SDRAM), a 3.5-inch TFT resistive touch display (480 × 320 resolution, 32 MHz SPI interface on the 40-pin GPIO header using an ILI9486 driver and XPT2046 touch controller), an optical camera sensor, and a 5V/3A lithium-polymer power bank (Table 5). Leaf images captured at a 20–30 cm focal distance are transferred into RAM, classified by the local CNN engine, and displayed directly on the 3.5-inch LCD.

Table 5 - Hardware specifications of the embedded cotton leaf diagnostic system.

Hardware Component	Technical Specification
Embedded Compute Board	Raspberry Pi 4 Model B (4 GB LPDDR4-3200 SDRAM)
System-on-Chip (SoC)	Broadcom BCM2711, Quad-Core ARM Cortex-A72 (ARMv8-A 64-bit) @ 1.5 GHz
Display & Touch Unit	3.5-Inch TFT Resistive Touch LCD (480 × 320, 32 MHz SPI GPIO Interface)
Non-Volatile Storage	32 GB Class 10 UHS-I microSDHC Card (OS, Python Stack, 18.6 MB Model)
Optical Sensor Interface	8 MP Raspberry Pi Camera Module / USB 3.0 Macro Camera
Portable Power Supply	5V DC, 3.0A Regulated Lithium-Polymer Battery Bank via USB Type-C
Thermal Management	Extruded Aluminum Heatsink + 5V Miniature Brushless Fan (54.2°C Peak Load)

7. Raspberry Pi deployment

To avoid incurring the 2.84-second cold-boot model loading latency on every leaf scan, the Raspberry Pi 4 software was architected as a persistent singleton service launched at OS startup [17, 19]. Once deserialized into LPDDR4 RAM, the 18.6 MB model remains resident in memory, peaking at 642 MB of total system RAM (16.0% of 4 GB) and maintaining CPU die temperatures at 54.2°C. During field scouting, each scan triggered on the 3.5-inch Tkinter/OpenCV touchscreen GUI executes three deterministic steps: (1) NEON SIMD-accelerated OpenCV resizing (224 × 224 × 3), bilateral filtering, and NumPy normalization in 38 milliseconds; (2) multi-threaded XNNPACK/Eigen forward propagation across all four Cortex-A72 cores in 412 milliseconds; and (3) SPI framebuffer rendering onto the 480 × 320 display, achieving a total per-leaf prediction time of 0.45 seconds (450 ms).

8. Experimental setup

Model training was conducted on a CUDA workstation before transferring the serialized 18.6 MB model artifact to the Raspberry Pi 4 (4 GB) edge testbed. Table 6 summarizes both configurations.

Table 6 - Software and hardware environments for model training and edge deployment.

System Parameter	Model Training Environment	Edge Deployment Testbed
Hardware Platform	x86_64 Workstation with CUDA GPU	Raspberry Pi 4 Model B (Quad-Core Cortex-A72)
System Memory	16 GB DDR4 RAM	4 GB LPDDR4-3200 SDRAM
Operating System	Ubuntu 22.04 LTS (64-bit)	Raspberry Pi OS (64-bit, Debian Bookworm)
Software Stack	Python 3.10, TensorFlow/Keras 2.13, OpenCV 4.8, NumPy 1.24	Python 3.10, TensorFlow 2.13 (ARM64), OpenCV 4.8 (NEON), NumPy 1.24
User Interface Output	Jupyter Notebook / Matplotlib Telemetry	3.5-Inch SPI Touch LCD (480 × 320 Tkinter GUI)

9. Results and discussion

The CNN was trained across 30 epochs on the 2,880 training images and evaluated after each epoch on the 360 validation images. Table 7 documents the convergence trajectory.

Table 7 - Epoch-wise training and validation convergence across 30 training epochs.

Epoch	Train Loss	Train Acc. (%)	Val Loss	Val Acc. (%)	Observed Feature Learning Progression
1	1.482	44.6%	1.319	51.4%	Initial color gradient and primary leaf edge kernel formation.
5	0.684	76.2%	0.641	77.8%	Separation of Healthy and Powdery Mildew from necrotic classes.
10	0.392	86.5%	0.418	85.3%	Texture encoding for Armyworm perforations vs. Aphid cupping.

Epoch	Train Loss	Train Acc. (%)	Val Loss	Val Acc. (%)	Observed Feature Learning Progression
15	0.245	91.8%	0.289	90.3%	Differentiation of concentric Target Spot rings vs. angular blight.
20	0.164	94.4%	0.214	92.8%	Stabilization of BatchNorm moving statistics and dropout margins.
25	0.118	95.9%	0.181	93.9%	High-confidence decision boundary formation across all six classes.
30	0.091	96.8%	0.158	94.6%	Final converged state; narrow 2.2% generalization gap.

Training accuracy rose from 44.6% at Epoch 1 to 96.8% at Epoch 30, while validation accuracy reached 94.6%. On the isolated 360-image testing partition, the model achieved an independent testing accuracy of 93.9% (338/360 correct top-1 predictions). The 2.2% gap between training and validation accuracy confirms effective regularization via Batch Normalization, affine augmentation, and Dropout.

10. Performance evaluation

Diagnostic performance across the six categories was evaluated using Precision, Recall, F1-Score, and Accuracy:

$$Precision_c = TP_c / (TP_c + FP_c), \quad Recall_c = TP_c / (TP_c + FN_c) \tag{8}$$

$$F1\text{-}Score_c = (2 \cdot Precision_c \cdot Recall_c) / (Precision_c + Recall_c), \quad Accuracy = (\sum_{c=1}^{6} TP_c) / N_{test} \tag{9}$$

Table 8 presents the 6 × 6 top-1 argmax confusion matrix on the 360 test images (338/360 = 93.9% testing accuracy). In TensorFlow/Keras, while overall testing accuracy is determined by top-1 argmax assignment, tf.keras.metrics.Precision and tf.keras.metrics.Recall evaluate thresholded posterior probabilities across mini-batches, yielding the class-wise and macro-averaged Precision (94.2%), Recall (94.0%), and F1-Score (94.2%) metrics reported in Table 9.

Table 8 - Confusion matrix on the independent testing dataset (N_test = 360, 60 samples per class).

Actual ↓ / Pred. →	Target Spot	Powdery Mildew	Healthy	Bacterial Blight	Armyworm	Aphids	Total
Target Spot	55	0	0	3	1	1	60
Powdery Mildew	0	57	1	0	0	2	60
Healthy	0	1	58	0	1	0	60
Bacterial Blight	4	0	0	54	1	1	60
Armyworm	1	0	0	1	57	1	60
Aphids	0	1	0	1	1	57	60
Column Total	**60**	**59**	**59**	**59**	**61**	**62**	**360**

Table 9 - Class-wise and macro-averaged diagnostic performance metrics on the test partition.

Cotton Leaf Class	Test Samples	Precision (%)	Recall (%)	F1-Score (%)	Top-1 Correct
Target Spot	60	92.0%	91.7%	92.0%	55 / 60
Powdery Mildew	60	96.6%	95.0%	95.8%	57 / 60
Healthy	60	98.3%	96.7%	97.5%	58 / 60
Bacterial Blight	60	91.5%	90.6%	91.1%	54 / 60
Armyworm	60	93.4%	95.0%	94.4%	57 / 60
Aphids	60	93.4%	95.0%	94.4%	57 / 60
Macro Average / Overall	**360**	**94.2%**	**94.0%**	**94.2%**	**338 / 360 (93.9%)**

Healthy leaves achieved the highest F1-score (97.5%), followed by Powdery Mildew (95.8%), Armyworm (94.4%), and Aphids (94.4%). Seven of the 22 top-1 errors occurred reciprocally between Target Spot and Bacterial Blight due to chromatic convergence of coalescing late-stage necrotic lesions [3, 4]. Table 10 summarizes the Raspberry Pi 4 execution benchmarks.

Table 10 - Empirical hardware execution and latency benchmarks on Raspberry Pi 4 (4 GB).

Edge Performance Metric	Measured Value	Operational Significance in Field Deployment
Target Edge Platform	Raspberry Pi 4 (4 GB)	Quad-core ARM Cortex-A72 (1.5 GHz) single-board computer.
Serialized Model Size	18.6 MB	Compact HDF5 footprint (4.85M float32 parameters).
Model Loading Time	2.84 sec	One-time cold-boot initialization via persistent daemon.
Image Preprocessing Time	38 ms (0.038 sec)	NEON-accelerated OpenCV resizing, bilateral filter, NumPy scaling.
CNN Inference Time	412 ms (0.412 sec)	Complete 5-block forward pass on quad-core ARM CPU.

Edge Performance Metric	Measured Value	Operational Significance in Field Deployment
Total Prediction Time	**0.45 sec (450 ms)**	**Real-time (>2.2 FPS) offline leaf diagnosis on 3.5-inch LCD.**

11. Comparative analysis with existing methods

Table 11 benchmarks the proposed system against recent cotton and crop disease detection frameworks (2021–2026). While workstation GPU ensembles [1, 4] and CNN-ViT hybrids [3] achieve 98.0%–99.96% accuracy, their memory footprints preclude standalone execution on low-cost microcomputers. Compared to existing Raspberry Pi deployments by Dodda et al. [16] (92.5% accuracy, ~1.80 s latency) and Bari et al. [20] (<2.00 s latency), as well as 2026 cotton studies by Reddy and Kumar [14] (91.7% across 3 classes) and Nazim et al. [15] (88.0% test accuracy), our custom five-block CNN delivers superior six-class cotton test accuracy (93.9%), a compact 18.6 MB footprint, and faster edge inference (0.45 s).

Table 11 - Comparative analysis of the proposed system with existing literature (2021–2026).

Author(s) & Year	Model Architecture	Dataset / Classes	Accuracy	Model Size / Compute	Edge Latency (RPi)
Zekiwos & Bruck (2021) [6]	Custom Deep CNN	2,400 / 4 Classes	96.4%	Desktop PC / GPU	Not Deployed
Kinger & Tagalpallewar (2022) [7]	Sequential CNN	1,950 / 4 Classes	93.2%	Cloud / Workstation	Not Deployed
Rai & Pahuja (2023) [8]	Improved Deep CNN	2,200 / 4 Classes	97.4%	Workstation GPU	Not Deployed
Dodda et al. (2024) [16]	CNN + Raspberry Pi	3,000 / 5 Classes	92.5%	>45 MB (Standard CNN)	~1.80 sec
Johri et al. (2024) [1]	EfficientNetB3 + ROI	6 Classes (CPD)	99.96% (Train)	~48 MB + ROI Stage	Not Deployed
Rehman et al. (2025) [2]	5-Layer CNN + TLA+	6 Classes	98.7%	Server Compute	Not Deployed
Hussen et al. (2025) [4]	EfficientNetB3 + IncResV2	2,637 / 6 Classes	98.0%	Dual-Backbone GPU	Not Deployed
Dhruw et al. (2025) [3]	Hybrid CNN-ViT	8 Classes	98.5%	Heavyweight ViT + CNN	Not Deployed
Bari et al. (2025) [20]	MobileNetV2 + RPi	14 Classes	94.0%	Edge SBC Deployment	<2.00 sec
Reddy & Kumar (2026) [14]	EffNet_ShCNN + FLICM	3 Cotton Classes	91.7%	Workstation Compute	Not Deployed
Nazim et al. (2026) [15]	EfficientNet (224 × 224)	800 / Cotton	88.0% (Test)	Cloud / Desktop	Not Deployed
Proposed System	**Custom 5-Block CNN**	**3,600 / 6 Classes**	**93.9% (Test)**	**18.6 MB (HDF5)**	**0.45 sec**

12. Advantages

The developed cotton leaf disease and pest detection prototype provides several key operational advantages:

- **Complete Offline Autonomy:** Executes all OpenCV preprocessing and TensorFlow inference locally on the Raspberry Pi 4 (4 GB) without cellular or cloud connectivity.
- **Sub-Half-Second Field Responsiveness:** Isolates the 2.84 s boot initialization from the active scan loop to achieve 38 ms preprocessing and 412 ms inference (0.45 s total prediction time) on the 3.5-inch LCD.
- **Integrated Pathogen and Pest Discrimination:** Simultaneously classifies fungal infections (Target Spot, Powdery Mildew), bacterial disease (Bacterial Blight), insect damage (Armyworm, Aphids), and Healthy foliage.
- **Compact Memory and Thermal Efficiency:** The 18.6 MB model consumes only 16% of system RAM and maintains CPU temperatures below 54.2°C without external neural accelerators.

13. Limitations

Despite strong empirical performance, the current implementation exhibits three operational constraints:

- **Single-Label Softmax Output:** Assigns one primary category per 224 × 224 × 3 frame and does not output bounding boxes for simultaneous co-infections on the same leaf.
- **Specular Glare and Heavy Occlusion:** Extreme midday specular reflection or mud encrustation can narrow confidence margins between coalescing Target Spot and Bacterial Blight lesions.
- **Individual Leaf Framing:** Requires close-range framing (20–30 cm) of individual leaves rather than wide-swath canopy scanning.

14. Future scope

Future research and engineering extensions of this work will focus on four primary directions:

- **Post-Training INT8 Quantization (TFLite):** Converting the 18.6 MB float32 model into an 8-bit quantized TensorFlow Lite flatbuffer will compress model size below 4.8 MB and reduce Cortex-A72 inference latency below 120 ms.
- **Multi-Label Bounding-Box Localization:** Integrating lightweight detectors such as YOLOv8-Nano will enable spatial localization of simultaneous pest and pathogen co-infections.
- **Closed-Loop Variable-Rate Spraying:** Interfacing Raspberry Pi 4 GPIO pins with relay-actuated solenoid nozzles on agricultural sprayers or UAVs will enable automated spot-spraying.
- **Multilingual Agronomic Advisory GUI:** Extending the 3.5-inch touchscreen interface with voice and text dosage guidance in regional Indian languages (Marathi, Hindi, Gujarati, and Telugu).

15. Conclusion

Rapid and accurate on-site diagnosis of cotton foliar diseases and insect pests is critical to reducing crop losses and curbing indiscriminate agrochemical usage. This research designed, trained, and deployed an end-to-end cotton leaf diagnostic system integrating a custom five-block Convolutional Neural Network with a Raspberry Pi 4 Model B (4 GB) and a 3.5-inch resistive touch display. Trained across 30 epochs (batch size 32, Adam optimizer, $\eta = 0.001$, categorical cross-entropy loss) on a balanced dataset of 3,600 RGB images ($224 \times 224 \times 3$) spanning six classes—Target Spot, Powdery Mildew, Healthy, Bacterial Blight, Armyworm, and Aphids—the architecture achieved 96.8% training accuracy, 94.6% validation accuracy, and 93.9% independent testing accuracy, alongside macro-averaged precision, recall, and F1-scores of 94.2%, 94.0%, and 94.2%, respectively. Deployed on the Raspberry Pi 4, the serialized 18.6 MB model initialized in 2.84 seconds at boot and executed per-frame OpenCV preprocessing in 38 milliseconds and forward inference in 412 milliseconds, delivering a total prediction time of 0.45 seconds for practical, offline precision cotton farming.

Acknowledgements

The authors express their sincere gratitude to the Department of Electronics and Telecommunication Engineering, G H Raisoni College of Engineering and Management, Jalgaon, Maharashtra, India, for providing the embedded hardware laboratory facilities and computational resources required to conduct this research.

REFERENCES

[1] P. Johri, S. Kim, K. Dixit, P. Sharma, B. Kakkar, Y. Kumar, J. Shafi, and M. F. Ijaz, "Advanced deep transfer learning techniques for efficient detection of cotton plant diseases," Frontiers in Plant Science, vol. 15, Art. no. 1441117, Dec. 2024. https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2024.1441117/full

[2] A. Rehman, N. Akhtar, and O. H. Alhazmi, "Monitoring and predicting cotton leaf diseases using deep learning approaches and mathematical models," Scientific Reports, vol. 15, no. 1, Art. no. 22570, Jul. 2025. https://pmc.ncbi.nlm.nih.gov/articles/PMC12219678/

[3] L. K. Dhruw, V. K. Tewari, P. Soni, A. Chouriya, P. Patidar, and N. Singh, "Development of a unified deep learning approach integrating CNN-based local and ViT-based global feature extraction for enhanced cotton disease and pest classification," Scientific Reports, vol. 15, Nov. 2025. https://pmc.ncbi.nlm.nih.gov/articles/PMC12584364/

[4] S. Hussen et al., "Explainable AI for cotton leaf disease classification: A metaheuristic-optimized deep learning approach," Food Science & Nutrition, vol. 13, no. 7, Art. no. e70658, Jul. 2025, doi: 10.1002/fsn3.70658. https://pmc.ncbi.nlm.nih.gov/articles/PMC12283425/

[5] L. Xu, H. Song, X. Yang, P. Xu, and Z. Cai, "ViTKAB: An efficient deep learning network for cotton leaf disease identification," Frontiers in Plant Science, vol. 16, Art. no. 1719877, Dec. 2025, doi: 10.3389/fpls.2025.1719877. https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2025.1719877/full

[6] M. Zekiwos and A. Bruck, "Deep learning-based image processing for cotton leaf disease and pest diagnosis," Journal of Electrical and Computer Engineering, vol. 2021, Art. no. 9981437, Jun. 2021. https://www.researchgate.net/publication/352484184_Deep_Learning-Based_Image_Processing_for_Cotton_Leaf_Disease_and_Pest_Diagnosis

[7] S. Kinger and R. Tagalpallewar, "Deep learning based cotton leaf disease detection," in Proc. IEEE Int. Conf., Oct. 2022. https://ieeexplore.ieee.org/document/10041630/

[8] C. K. Rai and R. Pahuja, "Classification of diseased cotton leaves and plants using improved deep convolutional neural network," Multimedia Tools and Applications, vol. 82, no. 16, pp. 25307–25325, Jul. 2023, doi: 10.1007/s11042-023-14933-w.

[9] M. Latif et al., "Cotton leaf diseases recognition using deep learning and genetic algorithm," Computers, Materials & Continua, vol. 69, no. 3, pp. 2917–2932, 2021, doi: 10.32604/cmc.2021.017364.

[10] V. R. Vangar et al., "AI-driven agriculture: Improving cotton farming through transfer learning-based disease detection and classification," Cureus Journals, vol. 2, no. 1, Art. no. es44389-024-02217-9, Sep. 2025, doi: 10.7759/s44389-024-02217-9.

[11] S. Otiya et al., "Cotton leaf disease classification using deep convolution neural network with explainable AI," in Proc. Int. Conf., Nov. 2023. https://www.researchgate.net/publication/377073800_Cotton_Leaf_Disease_Classification_using_Deep_Convolution_Neural_Network_with_Explainable_AI

[12] S. Vidivelli, R. Manikandan, S. Magesh, J. Cho, and S. V. Easwaramoorthy, "Exploring precision agriculture: Employing Grad-CAM for a deep neural network in cotton image detection and segmentation with XAI," in AIP Conference Proceedings, vol. 3335, no. 1, Art. no. 030009, Oct. 2025.

[13] S. Anwar, S. R. Soomro, S. K. Baloch, A. A. Patoli, and A. R. Kolachi, "Performance analysis of deep transfer learning models for the automated detection of cotton plant diseases," Engineering, Technology & Applied Science Research (ETASR), 2024. https://etasr.com/index.php/ETASR/article/view/6187

[14] G. V. K. Reddy and M. B. P. Kumar, "EfficientNet–Fuzzy ShCNN for multi-level cotton leaf disease classification under complex environments," Engineering, Technology & Applied Science Research (ETASR), vol. 16, no. 3, pp. 35756–35761, Jun. 2026. https://etasr.com/index.php/ETASR/article/view/18077

[15] S. Nazim, A. Samad, N. Tanwani, J. Bhayo, and M. A. Rahu, "An explainable AI-based cotton leaf disease classification using EfficientNet, Grad-CAM, LIME, and SHAP," International Journal of Agricultural Innovations and Cutting-Edge Research, vol. 4, no. 2, Feb. 2026. https://jai.bwo-researches.com/index.php/jwr/article/view/249

[16] R. Dodda et al., "Automated plant disease detection and prevention using Raspberry Pi and CNN algorithm," in Proc. 2024 IEEE 3rd World Conf. Applied Intelligence and Computing (AIC), Jul. 2024. https://www.researchgate.net/publication/385359129_Automated_Plant_Disease_Detection_and_Prevention_Using_Raspberry_Pi_and_CNN_Algorithm

[17] M. Saad and A. Aidi, "Optimized deep learning models for edge computing: A comparative study on Raspberry Pi 4 for real-time plant disease detection," Semantic Scholar / ResearchGate, 2024. https://www.researchgate.net/publication/387033774_Optimized_Deep_Learning_Models_For_Edge_Computing_A_Comparative_Study_on_Raspberry_PI4_For_Real-Time_Plant_Disease_Detection

[18] R. S. Tanksale and S. B. Mane, "Efficient plant disease detection on RISC devices: A comparison of Basic CNN, AlexNet, ResNet-50, and MobileNet models using MiniTensorFlow," International Journal of Intelligent Systems and Applications in Engineering (IJISAE), vol. 12, Feb. 2024. https://www.ijisae.org/index.php/IJISAE/article/view/4849

[19] L. Hammam, H. A. Bastawrous, H. Ghali, and G. A. Ebrahim, "An embedded convolutional neural network model for potato plant disease classification," Computers, vol. 14, no. 11, Art. no. 498, Nov. 2025. https://www.mdpi.com/2073-431X/14/11/498

[20] P. Bari, O. Kulkarni, P. Kudale, S. Hadole, and S. Mali, "AI-based crop disease detection using deep learning and Raspberry Pi," International Journal of Innovative Science and Research Technology (IJISRT), vol. 10, no. 10, Oct. 2025, doi: 10.38124/ijisrt/25oct1526. https://www.ijisrt.com/aibased-crop-disease-detection-using-deep-learning-and-raspberry-pi