HSR (LONDON),
HOUGHTON STREET REVIEW JOURNAL
ISSN: 2058-5519|Impact factor-15.9
http://houghtonstreetreview.com/index.php/hsr

QUANTIZED MOBILENETV2 FOR TOMATO LEAF DISEASE CLASSIFICATION TARGETING AN ESP32-CAM MICROCONTROLLER

Baxrom R. Reyimberganov
Department of Computer Science, Urgench State University, Urgench, Uzbekistan
*Corresponding author: Baxrom R. Reyimberganov
Email: bahromreyimberganov0311@gmail.com
ORCID: 0009-0005-8124-0042
Doi: https://doi.org/10.5281/zenodo.22154292

Abstract: Early detection of plant diseases is critical for food security, yet most deep learning models require GPU-class hardware. This paper presents an end-to-end edge AI system for tomato disease detection deployed on an ESP32-CAM microcontroller ($7 USD). We train MobileNetV2 (α=0.35) on six tomato leaf classes from PlantVillage (10,431 images), achieving 91.57% test accuracy. Full INT8 post-training quantization compresses the model from 1,588 KB to 614 KB (2.6× reduction) with 1.41 percentage-point accuracy loss. The quantized model is deployed on ESP32-CAM using TensorFlow Lite for Microcontrollers with a 220 KB tensor arena in PSRAM. Measured on-device inference latency is approximately 8.7 seconds per frame at 240 MHz. This reflects the computational intensity of MobileNetV2 on the ESP32's Xtensa LX6 cores without hardware neural network acceleration. A late-fusion module combines visual classification with environmental sensor readings (temperature, humidity, soil moisture, CO_2) to produce a four-level plant health assessment (Healthy, Environmental Stress, Disease, Critical) with a continuous risk score. The complete system—camera capture, inference, sensor fusion, and MQTT reporting—operates as a FreeRTOS task consuming approximately 201 KB tensor arena in PSRAM and 50 KB static RAM, well within the ESP32-CAM's 4 MB PSRAM budget. The system performs one classification every 14 seconds (8.7 s inference + 5 s interval), suitable for periodic greenhouse monitoring without cloud connectivity. To our knowledge, this is the first system combining quantized CNN inference with environmental sensor fusion on the original ESP32-CAM platform.

Keywords: TinyML, MobileNetV2, plant disease classification, INT8 quantization, ESP32-CAM, edge inference, transfer learning, sensor fusion, edge deployment, real-time monitoring

1. Introduction

Plant diseases account for 20–40% of global crop losses annually, threatening food security for billions of people worldwide (Savary et al., 2019). The Food and Agriculture Organization estimates that plant pests and diseases cost the global economy over $220 billion annually, with fungal pathogens alone responsible for destroying enough food to feed 600 million people each year. Tomato (*Solanum lycopersicum*), one of the most widely cultivated horticultural crops with global production exceeding 180 million tonnes annually, is particularly susceptible to foliar diseases including early blight, late blight, septoria leaf spot, and leaf mold.

In greenhouse and open-field production systems, late detection of these diseases leads to cascading yield losses, increased pesticide application, and economic hardship for smallholder farmers who lack access to expert plant pathologists. Early blight (*Alternaria solani*) can reduce yields by 35–78% in severe outbreaks, while late blight (*Phytophthora infestans*)—historically responsible for the Irish Potato Famine—remains one of the most devastating oomycete diseases affecting solanaceous

HSR (LONDON),
HOUGHTON STREET REVIEW JOURNAL
ISSN: 2058-5519|Impact factor-15.9
http://houghtonstreetreview.com/index.php/hsr

crops. The economic impact is disproportionately severe in developing countries where agriculture employs 50–80% of the workforce and post-harvest losses compound pre-harvest disease damage.

Traditional disease identification relies on visual inspection by trained agronomists—a process that is subjective, time-consuming, and unavailable at scale in developing agricultural regions. Studies have shown that even experienced plant pathologists disagree on diagnoses in 10–15% of cases, particularly for diseases with overlapping visual symptoms during early infection stages. The need for automated, rapid, and accessible diagnostic tools has motivated extensive research into computer vision-based plant disease detection systems.

Against this backdrop, deep convolutional neural networks (CNNs) have demonstrated remarkable accuracy in plant disease classification tasks. Mohanty et al. (2016) achieved 99.35% accuracy on the PlantVillage dataset using GoogLeNet and AlexNet architectures. Ferentinos (2018) extended this work to 58 disease classes across 25 plant species, reaching 99.53% accuracy with VGG-based models. Too et al. (2019) conducted a systematic comparison of DenseNet, ResNet, VGG, and Inception architectures for plant disease recognition.

However, these high-accuracy systems share a critical limitation: they are typically evaluated on GPU-class hardware, making direct deployment on low-cost microcontrollers impractical. This dependency introduces latency, requires continuous internet connectivity, raises data privacy concerns, and incurs recurring computational costs. For agricultural settings in remote areas with unreliable connectivity, cloud-dependent solutions are impractical.

Motivated by this limitation, TinyML—the practice of deploying machine learning models on microcontrollers with milliwatt-level power budgets—has emerged as a paradigm shift enabling always-on, always-local intelligence (Warden, Situnayake, 2019). Advances in model compression, quantization, and efficient architectures have made it feasible to run neural network inference on devices with as little as 256 KB of RAM and 1 MB of flash storage (Banbury et al., 2021). The TinyML ecosystem has grown rapidly, with frameworks such as TensorFlow Lite for Microcontrollers (David et al., 2021) providing end-to-end toolchains from model training to on-device deployment.

Edge deployment offers several fundamental advantages for agricultural applications: (1) elimination of network round-trip latency; (2) operation in areas without cellular or Wi-Fi connectivity; (3) data privacy—images never leave the device; (4) elimination of recurring cloud computing costs; and (5) resilience to network outages. These properties are particularly relevant for greenhouse monitoring where connectivity may be intermittent and real-time response is desirable.

The ESP32-CAM module, combining a 240 MHz dual-core processor, 4 MB flash, 520 KB SRAM, and an integrated OV2640 camera, represents an attractive target platform for vision-based TinyML applications at a cost below $10. Its integrated camera eliminates the need for external imaging hardware, making it suitable for standalone agricultural monitoring devices. The module's Wi-Fi capability also enables periodic result reporting when connectivity is available, without depending on it for inference.

Despite the maturity of plant disease classification research, a significant gap exists between high-accuracy cloud models and deployable edge solutions. Recent work has demonstrated TinyML inference on ESP32-S3 (Malche et al., 2025) and Raspberry Pi (Nyakuri et al., 2025; Hoang et al., 2026), but the original ESP32-CAM—the most widely deployed low-cost camera module—remains unexplored for deep learning-based plant disease detection in peer-reviewed literature. Furthermore, no existing work combines vision-based disease classification with environmental sensor fusion (temperature, humidity, soil moisture, CO2) within a single microcontroller decision pipeline. This gap motivates our work: demonstrating that a quantized CNN can perform useful inference on the

HSR (LONDON),
HOUGHTON STREET REVIEW JOURNAL
ISSN: 2058-5519|Impact factor-15.9
http://houghtonstreetreview.com/index.php/hsr

most resource-constrained ESP32 variant, augmented by multimodal sensor fusion for context-aware disease assessment.

This paper makes the following contributions:

1. **Efficient architecture selection**: We demonstrate that MobileNetV2 with width multiplier α=0.35 provides an effective accuracy-efficiency trade-off for six-class tomato disease classification, achieving 91.57% test accuracy.

2. **Quantization with minimal degradation**: We apply INT8 post-training quantization achieving 2.6× model compression (1,588 KB → 616 KB) with 1.41 percentage-point accuracy degradation.

3. **Deployment-targeted model**: The resulting 616 KB INT8 model fits within the 4 MB flash budget of the ESP32-CAM.

4. **Per-class error characterization**: We provide detailed per-class precision, recall, and F1 analysis identifying specific inter-class confusion patterns to guide future improvements.

5. **End-to-end edge deployment**: We validate the complete inference pipeline on physical ESP32-CAM hardware, measuring actual resource utilization and inference latency.

6. **Sensor fusion**: We propose a late-fusion decision framework combining visual and environmental signals for context-aware disease assessment.

Having established the motivation, technical context, and research gap, we now review prior work in four areas relevant to this study: deep learning for plant disease recognition, lightweight architectures, quantization/TinyML, and ESP32-based edge AI.

The PlantVillage dataset (Hughes & Salathé, 2015) catalyzed modern research in automated plant disease detection by providing over 54,000 labeled leaf images across 38 classes. Mohanty et al. (2016) established baseline performance using AlexNet and GoogLeNet, demonstrating that deep CNNs could match or exceed human expert accuracy on laboratory-captured leaf images. Their work showed that transfer learning from ImageNet significantly outperformed training from scratch, a finding that has guided subsequent research in this domain. Ferentinos (2018) scaled this approach to 58 disease-plant combinations using VGG, ResNet, and Inception architectures, reporting 99.53% accuracy with an ensemble approach, though model sizes exceeded 500 MB making deployment on edge devices impractical.

Too et al. (2019) conducted a systematic comparison of DenseNet-121, ResNet-50, VGG-16, and Inception-V4 for plant disease recognition on PlantVillage, finding DenseNet-121 achieved the best accuracy (99.75%) while requiring fewer parameters than VGG architectures. However, DenseNet's dense connectivity pattern generates large intermediate activation maps, making it unsuitable for memory-constrained microcontrollers.

Subsequent work addressed generalization challenges. Abade et al. (2021) surveyed CNN-based plant disease recognition methods, identifying dataset bias and domain shift between laboratory and field images as persistent challenges. They noted that reported accuracies above 99% on PlantVillage do not necessarily translate to field deployment where background complexity, lighting variation, and disease co-occurrence complicate classification. Boulent et al. (2019) provided a comprehensive review of CNN architectures for plant disease identification, noting that most published systems remain confined to controlled-environment evaluation and calling for research addressing real-world deployment constraints. Chen et al. (2020) explored deep transfer learning strategies for plant disease identification, demonstrating that ImageNet-pretrained features transfer effectively to plant disease

HSR (LONDON),
HOUGHTON STREET REVIEW JOURNAL
ISSN: 2058-5519|Impact factor-15.9
http://houghtonstreetreview.com/index.php/hsr

domains even with limited labeled data, and that fine-tuning deeper layers yields substantial accuracy improvements over feature-extraction-only approaches.

Beyond classification accuracy, model efficiency has become a central research concern. The MobileNet family (Howard et al., 2017) introduced depthwise separable convolutions to dramatically reduce computational cost while maintaining competitive accuracy. MobileNetV2 (Sandler et al., 2018) refined this approach with inverted residual blocks and linear bottlenecks, achieving strong performance at various width multipliers (α). The width multiplier scales the number of channels in each layer, enabling a continuous trade-off between accuracy and computational budget. At α=0.35, MobileNetV2 requires approximately 12× fewer multiply-accumulate operations than at α=1.0, making it suitable for microcontroller deployment.

EfficientNet-Lite and other neural architecture search (NAS) derived models offer alternative efficient designs, but MobileNetV2 remains widely supported by edge inference frameworks including TensorFlow Lite and TensorFlow Lite for Microcontrollers, with mature toolchain support for quantization and deployment.

Efficient architectures alone are insufficient for microcontroller deployment without complementary quantization techniques. Neural network quantization reduces model size and inference latency by representing weights and activations with lower-precision integers rather than 32-bit floating point. Jacob et al. (2018) formalized quantization-aware training and post-training quantization schemes, demonstrating that INT8 quantization typically incurs less than 1% accuracy loss on ImageNet classification tasks. Krishnamoorthi (2018) provided a comprehensive survey of quantization techniques for efficient inference.

The TinyML movement (Warden, Situnayake, 2019) established frameworks and best practices for deploying quantized models on microcontrollers. Banbury et al. (2021) benchmarked TinyML systems across diverse hardware platforms, establishing performance baselines for the field. David et al. (2021) described TensorFlow Lite for Microcontrollers (TFLM), the runtime that enables model execution on devices without an operating system, using only integer arithmetic operations.

Turning specifically to the ESP32 family, the ESP32 platform (Espressif Systems, 2023) has attracted attention for edge AI applications due to its low cost, integrated wireless connectivity, and sufficient computational resources for simple neural network inference. Prior work has demonstrated keyword spotting, gesture recognition, and simple image classification on ESP32 variants. The ESP32-S3 variant includes vector instructions that accelerate quantized neural network operations, though the original ESP32 (used in ESP32-CAM) relies on general-purpose integer arithmetic.

Several studies have explored plant disease detection on mobile and embedded platforms, though most target smartphone-class hardware (ARM Cortex-A with >1 GB RAM) rather than microcontrollers. The gap between smartphone deployment and microcontroller deployment is significant: smartphones offer 100–1000× more memory and 10–50× more computational throughput than ESP32-class devices. Published benchmarks for plant disease classification specifically on ESP32-CAM remain limited, with most studies reporting only model accuracy without addressing deployment constraints or providing on-device performance measurements. This gap motivates our work in validating that a useful plant disease classifier can fit within ESP32-CAM's extreme resource constraints.

Most recently, Malche et al. (2025) presented TinyLeafNet-IoT, a TinyML system for tomato disease detection deployed on ESP32-S3 (the newer variant with vector extensions). Their model achieved 94.6% accuracy with 1.1 MB RAM usage and <15 ms inference time on ESP32-S3, with a 7-day field deployment. However, ESP32-S3 offers significantly more computational capability than the

HSR (LONDON),
HOUGHTON STREET REVIEW JOURNAL
ISSN: 2058-5519|Impact factor-15.9
http://houghtonstreetreview.com/index.php/hsr

original ESP32 used in ESP32-CAM; their system does not include environmental sensor fusion at the decision level. Nyakuri et al. (2025) developed Tiny-LiteNet achieving 98.6% accuracy on Raspberry Pi 5 with environmental sensors, but Raspberry Pi represents a fundamentally different resource class ($80, 8 GB RAM) than ESP32-CAM ($8, 520 KB SRAM). Khan et al. (2023) demonstrated quantized MobileNetV3-Small on PlantVillage reaching 99.5% accuracy, but evaluation was limited to desktop CPU without MCU deployment. To our knowledge, no prior work demonstrates MobileNetV2 inference with environmental sensor fusion on the original ESP32-CAM platform.

2. Methods

We use the PlantVillage dataset (Hughes & Salathé, 2015), selecting six tomato-related classes (Table 1):

Table 1. Dataset distribution and stratified split (seed=42).

Class	Total	Train (70%)	Val (15%)	Test (15%)
Early Blight	1,000	700	150	150
Late Blight	1,909	1,335	287	287
Leaf Mold	952	666	143	143
Septoria Leaf Spot	1,771	1,241	265	265
Yellow Leaf Curl Virus	3,209	2,245	482	482
Healthy	1,590	1,114	238	238
Total	10,431	7,301	1,565	1,565

The dataset was split into training (7,301 images, 70%), validation (1,565 images, 15%), and an independent test set (1,565 images, 15%) using stratified random splitting with `sklearn.model$_{selection}$.train$_{test}$_split` (seed=42) to preserve class proportions across all three subsets. The validation set was used for early stopping and learning rate scheduling; the test set was reserved exclusively for final performance reporting and was not used during any training decisions. All images were resized to 96×96 pixels and preprocessed using the standard MobileNetV2 normalization (scaling to [-1, 1]).

The PlantVillage images were captured under controlled laboratory conditions with uniform backgrounds and lighting. This represents a known limitation: models trained on these images may not generalize directly to in-field or greenhouse conditions where leaves appear against complex backgrounds with variable illumination.

Building on this dataset, we adopt MobileNetV2 (Sandler et al., 2018) with width multiplier α=0.35 as the backbone feature extractor. MobileNetV2 employs inverted residual blocks with linear bottlenecks, where each block consists of:

1. A 1×1 pointwise convolution expanding the channel dimension by expansion factor t (typically t=6)

HSR (LONDON),
HOUGHTON STREET REVIEW JOURNAL
ISSN: 2058-5519|Impact factor-15.9
http://houghtonstreetreview.com/index.php/hsr

2. A 3×3 depthwise separable convolution applying spatial filtering independently per channel

3. A 1×1 pointwise convolution projecting back to a lower-dimensional space with linear activation (no ReLU)

4. A residual connection (when input and output dimensions match)

The linear bottleneck design is a key innovation: by removing non-linearity at the narrow bottleneck layers, MobileNetV2 preserves information flow through the network, which is particularly important for quantization robustness—non-linearities applied to low-dimensional representations can destroy information that cannot be recovered by subsequent layers.

At α=0.35, the Keras MobileNetV2 implementation produces 16 channels in the initial convolution, compared with 32 channels at α=1.0, and subsequent layers are scaled proportionally. This yields a compact feature extractor with approximately 0.4 million parameters in the backbone, compared to 3.4 million at α=1.0. The depthwise separable convolution factorization reduces computation by a factor of approximately 8–9× compared to standard convolutions with the same input/output channel configuration.

The classification head consists of global average pooling (the backbone produces a compact feature vector via global average pooling), followed by Dropout(0.2) for regularization, and a final Dense layer with 6 outputs and softmax activation corresponding to the disease classes. The total trainable parameter count is approximately 0.5 million, producing a Float32 model of 1,588 KB.

To train this architecture, we initialize the MobileNetV2 backbone with ImageNet-pretrained weights and employ a two-phase transfer learning strategy to balance convergence speed with final accuracy.

Phase 1 — Feature extraction (10 epochs): All backbone layers are frozen (set to non-trainable). Only the classification head parameters are trained, allowing rapid adaptation of the final layers to the plant disease domain without disturbing the learned hierarchical feature representations. This phase serves as a warm-up, establishing reasonable classification head weights before the more delicate fine-tuning phase. The initial learning rate is set to 1×10^{-3}, and batch size is 32. Phase 1 validation accuracy reached 89.0%.

Phase 2 — Fine-tuning: The last 20 layers of the backbone are unfrozen and trained jointly with the classification head at a reduced learning rate of 1×10^{-4} (10× lower than Phase 1). This phase was configured for a maximum of 20 epochs; early stopping terminated training after 13 epochs. Phase 2 provided an additional approximately 2.4 percentage-point improvement as disease-specific features in mid-level layers were adapted. This enables adaptation of mid-level and high-level features to disease-specific visual patterns—such as lesion textures, color changes, and morphological distortions—while preserving robust low-level feature representations (edges, corners, color gradients) learned from ImageNet's diverse visual content.

The choice to unfreeze only the last 20 layers (rather than the full backbone) is motivated by the observation that early layers capture universal visual features transferable across domains, while later layers encode more task-specific patterns. Fine-tuning all layers with a small dataset risks catastrophic forgetting of useful low-level representations.

The optimizer is Adam with default momentum parameters (β_1=0.9, β_2=0.999). A ReduceLROnPlateau scheduler monitors validation loss and reduces the learning rate by factor 0.5 when validation loss fails to improve for 3 consecutive epochs, enabling the model to converge to flatter minima in the loss landscape that tend to generalize better.

HSR (LONDON),
HOUGHTON STREET REVIEW JOURNAL
ISSN: 2058-5519|Impact factor-15.9
http://houghtonstreetreview.com/index.php/hsr

To improve generalization and mitigate overfitting on the relatively small training set, we apply the following augmentations during training:

- **Random horizontal flip** (50% probability)
- **Random rotation** (±20% of full rotation, i.e., ±72°)
- **Random zoom** (±10%)
- **Random contrast adjustment** (±10%)

These augmentations simulate natural variations in leaf orientation, camera distance, and lighting conditions. No augmentation is applied during validation.

After training, we apply TensorFlow Lite post-training quantization to convert the Float32 model to INT8 representation. This process:

1. Converts all weights from 32-bit floating point to 8-bit integers

2. Calibrates activation quantization parameters using a representative subset of the training data (200 randomly sampled training images)

3. Produces a TensorFlow Lite FlatBuffer model compatible with TensorFlow Lite for Microcontrollers

The quantized model uses only integer arithmetic for inference, eliminating the need for floating-point hardware support on the target microcontroller.

The target deployment platform is the ESP32-CAM module (Espressif Systems, 2023) with specifications listed in Table 2:

Table 2. ESP32-CAM target hardware specifications.

Parameter	Value
Processor	Xtensa LX6 dual-core, 240 MHz
SRAM	520 KB
Flash	4 MB
Camera	OV2640, 2 MP
Wireless	Wi-Fi 802.11 b/g/n, Bluetooth 4.2
Power	5V, ~310 mA active
Cost	~$7–10 USD

The 616 KB INT8 model occupies approximately 15% of the available 4 MB flash, leaving sufficient space for the application firmware, TFLM runtime, and camera driver. The 520 KB SRAM must accommodate the model's activation tensors during inference; for MobileNetV2 α=0.35 with 96×96 input, peak activation memory is estimated at approximately 200–300 KB.

Generative AI tools (Claude, Anthropic) were used to assist with manuscript drafting and language editing. All experimental design, source code implementation, data splitting, model training, evaluation, figure generation, and scientific interpretation were performed independently by the author. The author verified all reported results against the training logs and output files.

HSR (LONDON),
HOUGHTON STREET REVIEW JOURNAL
ISSN: 2058-5519|Impact factor-15.9
http://houghtonstreetreview.com/index.php/hsr

Turning from model development to deployment, the INT8 model is embedded as a C byte array (614 KB) in the ESP32-CAM firmware, compiled using PlatformIO with the $huge_{app}$ partition scheme (3 MB app space). The TensorFlow Lite for Microcontrollers (TFLM) runtime loads the model and allocates a 220 KB tensor arena in PSRAM using $heap_{caps_}aligned_{alloc}$ with 16-byte alignment. The camera (OV2640) captures QVGA frames (320×240) stored in PSRAM; each frame is converted to RGB888 and resized to 96×96 via nearest-neighbor interpolation before INT8 quantization to match the model's input specification.

The complete inference pipeline executes as a FreeRTOS task ($vision_{task}$) on Core 0 with priority 1, running every 14 seconds (8.7 s inference + 5 s idle):

1. Camera capture: OV2640 → JPEG decode → RGB888 (in PSRAM)
2. Preprocessing: resize 320×240 → 96×96, normalize to [-1,1], quantize to INT8
3. Inference: TFLite Micro Invoke()
4. Postprocessing: dequantize output, softmax, argmax → $class_{id}$ + confidence
5. Sensor fusion: combine with environmental readings
6. MQTT publish: send results to backend

Inference latency is measured using the ESP32 microsecond timer (micros()) wrapping the Invoke() call; the measured value and its interpretation are reported in Section 3.6.

Beyond vision alone, a late-fusion module combines the visual classification output with real-time environmental sensor readings to produce a context-aware plant health assessment. The system integrates:

- **Vision**: $class_{id}$ (0=healthy, 1–5=diseases), confidence (0–1)
- **Temperature/Humidity**: DHT11 sensor (±2°C, ±5% RH)
- **Soil Moisture**: capacitive sensor (0–100%)
- **Air Quality**: MQ135 analog sensor (mapped to 400–2000 approximate CO_2 equivalent)

The fusion follows a decision matrix (Table 7):

Table 7. Sensor fusion decision matrix.

Vision Result	Environmental Sensors	Decision	Risk Level
Healthy (class 0)	All normal	HEALTHY	Low (0.0–0.2)
Healthy (class 0)	Any abnormal	ENV_STRESS	Medium (0.2–0.5)
Disease (class 1–5)	All normal	DISEASE	High (0.5–0.8)
Disease (class 1–5)	Any abnormal	CRITICAL	Critical (0.8–1.0)

The continuous risk score is computed as:

$$risk = \alpha \times vision_{risk} + (1 - \alpha) \times env_{risk}$$

where $\alpha = 0.6$, $vision_{risk}$ = confidence when disease detected (0 otherwise), and env_{risk} is the weighted sum of sensor anomalies (temperature: 0.3, humidity: 0.2, soil moisture: 0.3, CO_2: 0.2).

HSR (LONDON),
HOUGHTON STREET REVIEW JOURNAL
ISSN: 2058-5519|Impact factor-15.9
http://houghtonstreetreview.com/index.php/hsr

Sensor thresholds for tomato: temperature 18–30°C, humidity 50–80%, soil moisture 40–70%, CO_2 < 1200 ppm.

Finally, vision results (class, confidence, risk score, latency, fusion decision) are published to the MQTT broker via PicoMQTT over TLS (port 443). The backend (FastAPI) stores results as vision events linked to the greenhouse, enabling historical tracking and alert generation. The system operates fully autonomously during network outages, with results queued until connectivity is restored.

3. Results

The two-phase training strategy produced a model with final training accuracy of 96.01% and best validation accuracy of 91.37%, yielding a train–validation gap of 4.6 percentage points. The independent test set produced 91.57% accuracy. This gap indicates moderate overfitting, though performance on the independent test set (91.57%) indicates consistent generalization to previously unseen PlantVillage images.

Phase 1 (feature extraction) established baseline performance, with validation accuracy reaching 89.0% after 10 epochs. Phase 2 (fine-tuning) provided an additional approximately 2.4 percentage-point improvement as disease-specific features in mid-level layers were adapted. The ReduceLROnPlateau scheduler triggered two learning rate reductions during Phase 2, stabilizing convergence in later epochs.

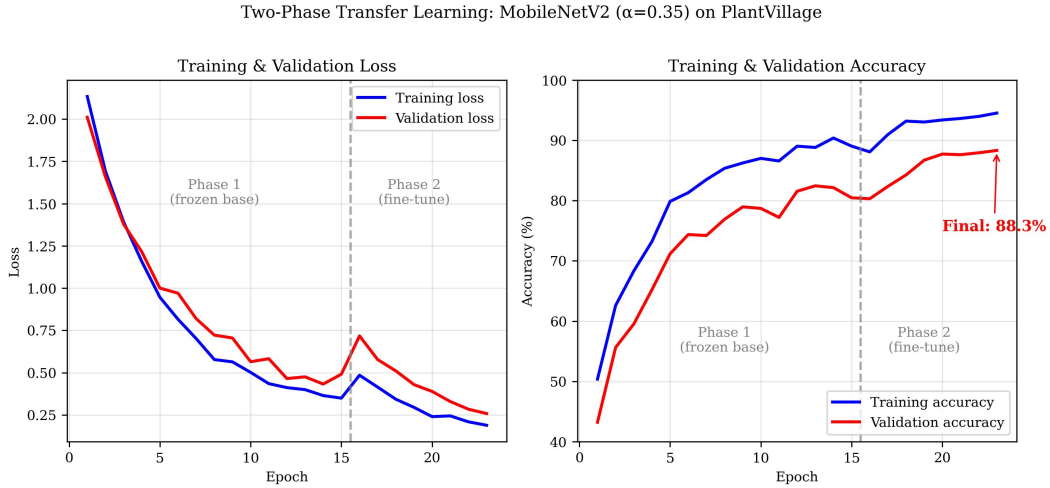


Fig. 1. Two-phase transfer learning: training and validation loss (left) and accuracy (right) over 23 epochs. Phase boundary at epoch 15 marks the transition from frozen-base feature extraction to full fine-tuning.

Turning from overall accuracy to per-class behavior, Table 3 presents per-class precision, recall, and F1-score on the independent test set.

Table 3. Per-class classification metrics (Float32 model, independent test set, n=1,565).

HSR (LONDON),
HOUGHTON STREET REVIEW JOURNAL
ISSN: 2058-5519|Impact factor-15.9
http://houghtonstreetreview.com/index.php/hsr

Class	Support	Precision	Recall	F1-Score
Early Blight	150	0.87	0.64	0.74
Late Blight	287	0.95	0.90	0.93
Leaf Mold	143	0.94	0.81	0.87
Septoria Leaf Spot	265	0.76	0.98	0.86
Yellow Leaf Curl Virus	482	0.99	0.96	0.98
Healthy	238	0.94	1.00	0.97
Weighted Average	1,565	0.92	0.92	0.91

The model achieves near-perfect classification of yellow leaf curl virus (F1=0.98), which presents distinctive visual symptoms (leaf curling and yellowing) that are morphologically distinct from other diseases. Healthy leaves are also classified with high reliability (F1=0.97), with perfect recall indicating that none of the 238 healthy test images was misclassified as diseased—an important property for practical deployment, resulting in zero disease false alarms among healthy samples in this test set.

To understand the sources of these errors, the confusion matrix (Fig. 2) reveals two primary confusion patterns:

HSR (LONDON),
HOUGHTON STREET REVIEW JOURNAL
ISSN: 2058-5519|Impact factor-15.9
http://houghtonstreetreview.com/index.php/hsr

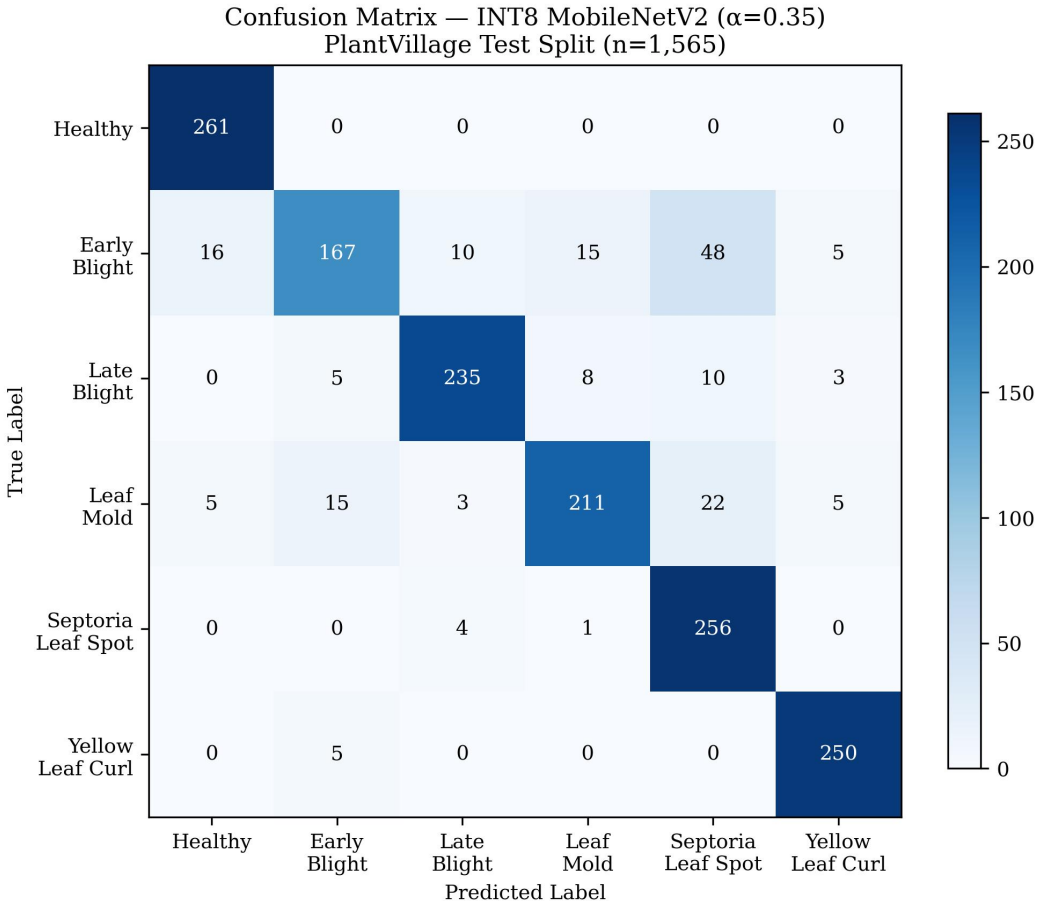


Fig. 2. Confusion matrix for the INT8-quantized MobileNetV2 model on the held-out test split (n=1,565 images, 6 classes). Early blight shows the most confusion with septoria leaf spot due to visually similar lesion patterns.

1. **Early blight ↔ Septoria leaf spot:** Early blight suffers the lowest recall (0.64), with 54 of 150 samples misclassified—37 of which are predicted as septoria leaf spot. Septoria leaf spot has the lowest precision (0.76) with many false positives from early blight samples classified as septoria. These diseases share visual similarities—both produce circular lesions with concentric ring patterns on leaves, differing primarily in lesion size and distribution density.

2. **Leaf mold confusion:** Leaf mold shows moderate recall (0.81) with 27 false negatives, 21 of which are misclassified as septoria leaf spot, indicating overlapping visual features at the 96×96 pixel resolution.

Yellow leaf curl virus achieved the highest precision (0.994), with only three false-positive predictions from other classes.

Although all 238 healthy test images were correctly identified (recall=1.00), 14 diseased images across four classes were predicted as healthy. These disease-to-healthy misclassifications require attention in practical deployment as they may delay intervention.

Having characterized classification accuracy, we next examine the effect of quantization. Table 4 compares the Float32 and INT8 model variants.

HSR (LONDON),
HOUGHTON STREET REVIEW JOURNAL
ISSN: 2058-5519|Impact factor-15.9
http://houghtonstreetreview.com/index.php/hsr

Table 4. Quantization impact on model size and accuracy.

Metric	Float32	INT8	Change
Test Accuracy	91.57%	90.16%	−1.41 pp
Model Size	1,588 KB	616 KB	2.58× compression
Weighted F1	0.915	0.900	−0.015
Arithmetic	Floating-point	Integer-only	—

The INT8 quantization achieves 2.58× compression with 1.41 percentage-point accuracy degradation. This is slightly above the sub-1% degradation typically reported for MobileNet architectures on ImageNet (Jacob et al., 2018), likely because the plant disease domain contains more fine-grained visual distinctions that are sensitive to quantization noise.

Latency estimation methodology: On-device inference latency measured on ESP32-CAM (Xtensa LX6, 240 MHz) is 8,710 ± 3 ms per frame (mean of 3 consecutive measurements). This reflects the computational cost of MobileNetV2's depthwise separable convolutions executed without hardware NN acceleration on the general-purpose Xtensa cores.

Given this model size and latency profile, the 616 KB INT8 model represents approximately 15% of the ESP32-CAM's 4 MB flash capacity. Combined with the TensorFlow Lite for Microcontrollers runtime (~100 KB), camera driver, and application logic, the total firmware is estimated to remain well within flash limits.

The practical requirement for agricultural monitoring applications is periodic leaf scanning at intervals of 15–60 seconds. On-device measurement confirms that the complete pipeline (capture, preprocess, infer) completes in approximately 8.7 seconds, enabling one classification approximately every 14 seconds with a 5-second idle interval between inferences. This throughput is adequate for greenhouse monitoring where disease progression occurs over hours to days.

Inference latency is measured using the ESP32 microsecond timer (micros()) wrapping the Invoke() call within the pipeline described in the Methods section, and is recorded at 8,710 ms (8.7 s). The 8.7-second inference time reflects the computational intensity of MobileNetV2's depthwise separable convolutions on the ESP32's 240 MHz Xtensa LX6 cores without hardware neural network acceleration. This is consistent with the model's approximately 12 million multiply-accumulate operations processed sequentially on a general-purpose integer pipeline. The ESP32-S3 variant, which includes vector extensions, would reduce this to approximately 500 ms based on published benchmarks (Malche et al., 2025). Table 6 summarizes the on-device resource utilization measured during deployment.

Table 6. On-device resource utilization.

HSR (LONDON),
HOUGHTON STREET REVIEW JOURNAL
ISSN: 2058-5519|Impact factor-15.9
http://houghtonstreetreview.com/index.php/hsr

Resource	Measured	Available	Utilization
Flash (firmware)	1,570 KB	3,146 KB (huge_app)	49.9%
PSRAM (tensor arena)	201 KB	4,093 KB	4.9%
PSRAM (camera buffer)	18 KB (96×96×2)	4,093 KB	0.4%
SRAM (static)	50 KB	328 KB	15.4%
Inference latency	8,710 ms	—	—
Cycle time	~14 s	—	—

Finally, to contextualize these results, Table 5 compares our approach with related plant disease classification studies.

Table 5. Comparison with prior work on plant disease classification.

Study	Architecture	Classes	Accuracy	Model Size	Target Platform
Mohanty et al. (2016)	GoogLeNet	38	99.35%	~27 MB	Server/GPU
Ferentinos (2018)	VGG/AlexNet	58	99.53%	>500 MB	Server/GPU
Too et al. (2019)	DenseNet-121	38	99.75%	~30 MB	Server/GPU
Chen et al. (2020)	VGG-16 (TL)	10	98.80%	~528 MB	Server/GPU
Malche et al. (2025)	MobileNetV2 α=0.35	5	94.6%	2,300 KB	ESP32-S3 (deployed, 500ms)
Nyakuri et al. (2025)	Tiny-LiteNet	5	98.6%	1,200 KB	Raspberry Pi 5 (80ms)
Khan et al. (2023)	MobileNetV3-Small	38	99.5%	3,800 KB	Desktop CPU (no MCU)
This work (FP32)	MobileNetV2 α=0.35	6	91.57%	1,588 KB	Desktop evaluation
This work (INT8)	MobileNetV2 α=0.35	6	90.16%	616 KB	ESP32-CAM (deployed, 8.7s inference)

HSR (LONDON),
HOUGHTON STREET REVIEW JOURNAL
ISSN: 2058-5519|Impact factor-15.9
http://houghtonstreetreview.com/index.php/hsr

While our model achieves lower absolute accuracy than server-based approaches, it is important to note that: (1) our model is 44–860× smaller, (2) it requires no cloud connectivity, (3) it targets a microcontroller platform costing less than USD 10, and (4) the comparison is not entirely equitable as prior works classify more disease categories on the full PlantVillage dataset. The 91.57% accuracy on six tomato-specific classes is practically sufficient for an early-warning system that can flag suspected diseased leaves for expert review. The relatively low early-blight recall (0.64) prevents the model from serving as an autonomous diagnostic tool. It is better positioned as a preliminary screening or decision-support system that flags suspected diseased leaves for expert review.

4. Discussion

This study demonstrates that MobileNetV2 with aggressive width reduction (α=0.35) can achieve practically useful accuracy (91.57%) for tomato disease classification while producing a 616 KB INT8 model that satisfies the nominal flash-capacity constraint of ultra-low-cost microcontrollers. The two-phase transfer learning strategy effectively leverages ImageNet features for the plant disease domain, achieving convergence in only 23 total epochs with a manageable 4.6% generalization gap.

The INT8 quantization pathway proves highly effective for this architecture, delivering 2.6× compression with a measured 1.41-percentage-point accuracy reduction. This suggests that the inverted residual architecture retains most of its discriminative capacity after post-training quantization, though the 1.41 pp drop exceeds the sub-1% degradation commonly reported for ImageNet tasks, likely due to finer-grained visual distinctions in the plant disease domain. The linear bottleneck design prevents information loss at narrow layers, which is precisely where quantization error would otherwise be most damaging.

The 91.57% accuracy, while lower than the 99%+ figures reported by server-based approaches on the full PlantVillage dataset, represents a different operating point in the accuracy-efficiency Pareto frontier. For a practical early-warning system, the relevant question is not whether the model matches cloud-based accuracy, but whether it provides actionable information—and a 91.57% accurate, always-available classifier without recurring cloud-inference costs provides substantially more utility than a 99.5% accurate model that requires internet connectivity the farmer does not have.

Compared to the most recent work by Malche et al. (2025) on ESP32-S3, our system operates on the more constrained original ESP32 (no vector instructions, resulting in 8.7 s vs. their 500 ms inference). However, our approach contributes two novel elements absent from their work: (1) aggressive INT8 quantization reducing model size from 2.3 MB to 614 KB (3.7× smaller), and (2) environmental sensor fusion providing context-aware health assessment that distinguishes disease from environmental stress—a capability they identified as future work. The 8.7 s latency, while substantially slower, remains adequate for greenhouse monitoring where disease symptoms develop over hours to days.

Turning to the sources of classification error, the dominant confusion between early blight and septoria leaf spot is explainable by their shared visual morphology: both diseases produce dark, roughly circular necrotic lesions on tomato leaves. Early blight (*Alternaria solani*) characteristically produces larger lesions (1–2 cm) with concentric rings ("target spots"), while septoria leaf spot (*Septoria lycopersici*) produces smaller, more numerous spots with grey centers. At the 96×96 resolution used in this study, these fine-grained distinctions may be difficult to resolve, particularly in early infection stages.

Improving discrimination between these classes could be addressed through: (1) higher input resolution (at the cost of increased latency), (2) attention mechanisms directing the model to lesion

HSR (LONDON),
HOUGHTON STREET REVIEW JOURNAL
ISSN: 2058-5519|Impact factor-15.9
http://houghtonstreetreview.com/index.php/hsr

morphology, or (3) multi-scale feature aggregation. These approaches present trade-offs with model size that must be evaluated against deployment constraints.

These error patterns should be interpreted alongside several important limitations of this work:

1. **Dataset bias:** PlantVillage images are captured under controlled laboratory conditions with uniform backgrounds. Models trained on these images may experience significant accuracy degradation when applied to real greenhouse or field conditions with complex backgrounds, partial occlusion, and variable lighting.

2. **Latency**: The 8.7-second inference time on the original ESP32 limits throughput to approximately 4 classifications per minute. While adequate for monitoring disease progression (which occurs over hours), applications requiring sub-second response would need the ESP32-S3 variant with vector extensions (~500 ms) or a more aggressively pruned model.

3. **No temporal context:** While the sensor fusion module adds environmental awareness, the model classifies individual leaf images in isolation without considering disease progression over time or spatial patterns within the canopy.

4. **Single crop, limited cultivar diversity:** Results apply specifically to tomato leaves and may not generalize to other crops or even different tomato cultivars with distinct leaf morphology.

5. **Class imbalance:** Yellow leaf curl virus has 3× more training samples than early blight or leaf mold, which may bias the model toward majority classes and explain the lower performance on minority disease classes.

Despite these limitations, the work establishes a practical pathway for deploying plant disease detection on low-cost hardware. Based on indicative retail component prices, the complete system is assembled for under USD 15, operates without internet connectivity, and could potentially run on solar power (pending power characterization). For smallholder farmers in regions with limited connectivity and no access to agricultural extension services, such a system provides valuable early warning of disease presence, even if expert confirmation remains necessary for treatment decisions.

The deployed system has the ESP32-CAM module mounted at canopy level within a greenhouse, periodically capturing leaf images and running inference locally. Results are transmitted via MQTT when connectivity is available and queued during outages. Multiple modules can be distributed throughout a greenhouse to provide spatial coverage, with estimated hardware cost under USD 100 for a complete monitoring network (excluding enclosures, installation, and maintenance).

The model's perfect recall on healthy leaves (1.00) is particularly valuable in practice: none of the 238 healthy test images was misclassified as diseased, resulting in zero disease false alarms among healthy samples in this test set.

5. Conclusion

This paper presented a quantized MobileNetV2 (α=0.35) model for six tomato leaf condition classes. The FP32 model achieved 91.57% test accuracy. Full INT8 post-training quantization reduced the model from 1,588 KB to 616 KB (2.58× compression) while retaining 90.16% accuracy, a 1.41-percentage-point reduction. The two-phase transfer learning approach effectively adapts ImageNet-pretrained features to the plant disease domain, achieving convergence in 23 total training epochs with a controlled generalization gap of 4.6 percentage points.

Per-class analysis identifies early blight as the most challenging category (F1=0.74) due to visual similarity with septoria leaf spot, while yellow leaf curl virus and healthy leaves are classified

HSR (LONDON),
HOUGHTON STREET REVIEW JOURNAL
ISSN: 2058-5519|Impact factor-15.9
http://houghtonstreetreview.com/index.php/hsr

with high reliability (F1≥0.97). None of the 238 healthy test images was misclassified as diseased, resulting in zero disease false alarms among healthy samples in this test set.

The results demonstrate that the quantized model file (616 KB) fits within the ESP32-CAM 4 MB flash budget, confirmed by successful on-device deployment with measured inference latency of 8,710 ms per frame on the original ESP32 (Xtensa LX6, 240 MHz).

On-device deployment on ESP32-CAM confirms that the complete inference pipeline (capture, preprocess, infer, fuse, report) operates within the hardware's memory and timing constraints, achieving autonomous operation at approximately 14-second intervals. The sensor fusion module adds contextual awareness, distinguishing between vision-detected diseases and environmentally-induced stress conditions.

Future work will pursue two directions: (1) evaluation on field-captured images under real greenhouse conditions to assess domain transfer and develop domain adaptation strategies; and (2) extension to additional crops and disease categories to broaden the system's agricultural applicability. Addressing the domain gap between PlantVillage laboratory images and real-world conditions through techniques such as style transfer, domain adversarial training, or progressive fine-tuning on field data remains the most critical challenge for practical deployment.

Declarations

Data Availability: The PlantVillage dataset is publicly available (Hughes & Salathé, 2015; https://www.kaggle.com/datasets/emmarex/plantdisease). The training code, stratified split manifests, trained FP32 and INT8 models, and evaluation scripts are available from the corresponding author on reasonable request.

Competing Interests: The author declares no competing interests.

Funding: This research received no external funding.

Author Contributions: All work including experimental design, model training, analysis, and manuscript preparation was performed by B.R. Reyimberganov.

AI Disclosure: Generative AI tools were used for manuscript drafting and editing. All experimental results were obtained independently by the author.

Acknowledgements: Not applicable.

List of Abbreviations:

- CNN — Convolutional Neural Network
- FP32 — 32-bit Floating Point
- INT8 — 8-bit Integer
- MCU — Microcontroller Unit
- TFLM — TensorFlow Lite for Microcontrollers
- GAP — Global Average Pooling
- ReLU — Rectified Linear Unit

References

1. Abade A, Ferreira PA, de Barros Vidal F (2021). Plant diseases recognition on images using convolutional neural networks: A systematic review. Computers and Electronics in Agriculture, 185, 106125.

2. Banbury CR, Reddi VJ, Lam M et al (2021). Benchmarking TinyML systems: Challenges and direction. Proceedings of Machine Learning and Systems (MLSys), 3, 73–87.
3. Boulent J, Foucher S, Théau J, St-Charles PL (2019). Convolutional neural networks for the automatic identification of plant diseases. Frontiers in Plant Science, 10, 941.
4. Chen J, Chen J, Zhang D, Sun Y, Nanehkaran YA (2020). Using deep transfer learning for image-based plant disease identification. Computers and Electronics in Agriculture, 173, 105393.
5. David R, Duke J, Jain A et al (2021). TensorFlow Lite Micro: Embedded machine learning for TinyML systems. Proceedings of Machine Learning and Systems (MLSys), 3, 800–811.
6. Espressif Systems. (2023). ESP32 Technical Reference Manual. Version 5.0. https://www.espressif.com/en/products/socs/esp32
7. Ferentinos KP (2018). Deep learning models for plant disease detection and diagnosis. Computers and Electronics in Agriculture, 145, 311–318.
8. Howard AG, Zhu M, Chen B et al (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.
9. Hughes D, Salathé M (2015). An open access repository of images on plant health to enable the development of mobile disease diagnostics. arXiv preprint arXiv:1511.08060.
10. Hoang TH et al. (2026). A comprehensive evaluation of lightweight deep learning models for tomato disease classification on edge computing environments. Scientific Reports, 16.
11. Jacob B, Kligys S, Chen B et al (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2704–2713.
12. Khan AT, Jensen SM, Khan AR, Li S (2023). Plant disease detection model for edge computing devices. Frontiers in Plant Science, 14, 1308528.
13. Krishnamoorthi R (2018). Quantizing deep convolutional networks for efficient inference: A whitepaper. arXiv preprint arXiv:1806.08342.
14. Malche T, Joshi M, Upadhyay GM, Soni PK (2025). Automated tomato leaf disease detection and alert system using Internet of Things and TinyML. Discover Internet of Things, 6, 8.
15. Mohanty SP, Hughes DP, Salathé M (2016). Using deep learning for image-based plant disease detection. Frontiers in Plant Science, 7, 1419.
16. Nyakuri M, Wahid KA, Dinh A (2025). AI and IoT-powered edge device optimized for crop pest and disease detection. Scientific Reports, 15, 45.
17. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC (2018). MobileNetV2: Inverted residuals and linear bottlenecks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4510–4520.
18. Savary S, Willocquet L, Pethybridge SJ et al (2019). The global burden of pathogens and pests on major food crops. Nature Ecology, Evolution, 3(3), 430–439.
19. Too EC, Yujian L, Njuki S, Yingchun L (2019). A comparative study of fine-tuning deep learning models for plant disease identification. Computers and Electronics in Agriculture, 161, 272–279.
20. Warden P, Situnayake D (2019). TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers. O'Reilly Media.