electronics

MDPI

Article

An Efficient Attention-Enhanced MobileNetV2 Framework for Plant Disease Detection on Resource-Constrained Devices

Emmanuel Udoh, Mohammed Ayoub Alaoui Mhamdi * and Madjid Allili

Department of Computer Science, Bishop's University, Sherbrooke, QC J1M 1Z7, Canada; eudoh23@ubishops.ca (E.U.); mallili@ubishops.ca (M.A.)
* Correspondence: malaoui@ubishops.ca

Abstract

Leaf disease diagnosis needs models that are accurate enough for agronomic use yet small enough for constrained computing settings. This study examines a late-attention MobileNetV2 design in which one Convolutional Block Attention Module (CBAM) is inserted between the last MobileNetV2 convolutional map and global average pooling. The experiments use 54,306 controlled-background PlantVillage images spanning 38 classes. Under a uniform saved-model re-evaluation, MobileNetV2 + CBAM obtained 97.17% accuracy and 97.15% weighted F1-score, whereas MobileNetV2 obtained 96.78% and 96.73%. On the converted models, paired testing gave a 0.64-percentage-point accuracy advantage for the CBAM variant (95% CI: 0.31–0.96; exact McNemar $p < 0.001$). The proposed network has 4.02 million parameters, costs 0.604 GFLOPs (about 0.302 GMACs), and yields a 4.07 MiB dynamic-range-quantized TensorFlow Lite file with 96.70% accuracy. Batch-one inference on an Intel i7-11800H CPU with TensorFlow Lite/XNNPACK and eight threads reached a median of 45.42 ms (P95: 102.54 ms), excluding preprocessing. Grad-CAM inspection illustrates both lesion-centered activation and unresolved shared errors. The evidence therefore supports a compact accuracy–cost compromise for the tested conditions, while field robustness, energy use, repeated training runs, and target-device behavior remain open validation requirements.

Keywords: plant disease detection; MobileNetV2; CBAM; lightweight deep learning; edge computing; TensorFlow Lite; precision agriculture

1. Introduction

Plant-pathogen damage continues to reduce crop yield and farmer income, and leaf symptoms are often among the first visible indicators of infection. In practice, diagnosis still frequently depends on human inspection, which can vary with the observer's experience, local access to agronomists, and the time available for field monitoring. These constraints are especially problematic in regions where specialist support and computing resources are limited. An image-based diagnostic tool therefore needs to be not only accurate, but also reproducible, scalable, and realistic for low-resource agricultural workflows.

Deep convolutional learning has made leaf-image classification far less dependent on manually designed color, texture, or shape descriptors. CNNs learn visual representations directly from data, and public resources such as PlantVillage have made it easier to compare disease-recognition models under shared image collections [1,2]. High test accuracy alone, however, is insufficient for eventual farm-facing use. Architectures with many parameters

Academic Editor: George A. Tsihrintzis

Received: 1 June 2026
Revised: 22 July 2026
Accepted: 23 July 2026
Published: 28 July 2026

or expensive convolutional operations may be difficult to store, convert, or run with acceptable latency on mobile and embedded hardware.

MobileNetV2 is attractive in this setting because depthwise separable convolutions, inverted residual connections, and linear bottlenecks reduce the computational burden of feature extraction [3]. A compact backbone can nevertheless discard useful detail when symptoms are small, irregular, or visually similar across disease classes. Attention modules provide one possible remedy by reweighting learned feature maps rather than replacing the backbone with a much larger network. CBAM is considered here because its channel branch selects informative feature responses and its spatial branch reweights their locations [4].

This paper evaluates an attention-enhanced MobileNetV2 framework as a candidate for resource-constrained plant disease classification. CBAM is integrated after the final MobileNetV2 convolutional feature map and before global average pooling. The study does not claim that combining two established components is itself a new attention mechanism. Instead, it defines and evaluates a deliberately late-attention placement: the attention block is applied once to a compact, semantically rich feature tensor rather than repeatedly inside early or middle backbone stages, where it would act on larger maps and increase memory traffic. This differs from MobileNetV2-attention designs that insert attention throughout the feature extractor or attach attention after feature pooling, although the present experiment does not prove the late placement is globally optimal. The model is evaluated on PlantVillage against MobileNetV2, InceptionV3, and Xception using classification and computational proxy metrics. Because PlantVillage uses controlled backgrounds and latency was not measured on a target edge device, the evaluation is evidence of compactness under the measured conditions rather than proof of field or real-time deployment readiness.

The main contributions of this paper are summarized as follows:

- A reproducible late-attention configuration is specified by placing CBAM once, after MobileNetV2's final convolutional feature map and before global average pooling, so attention operates on a compact semantic tensor rather than on larger early feature maps.
- The incremental effect of this configuration is quantified against an otherwise comparable MobileNetV2 baseline.
- A comparative evaluation is conducted against MobileNetV2, InceptionV3, and Xception using standard classification metrics, including support-weighted and macro-averaged summaries, on the PlantVillage dataset.
- The study reports parameter count, FLOPs/MACs, serialized model size, dynamic-range TFLite accuracy, training time, and standardized laptop-CPU latency, while separating these platform-specific measurements from target-device claims.
- The limitations imposed by controlled-background data, a single training run, partially fine-tuned backbones, legacy generator subsampling, audited cross-split duplicates, missing placement ablations, and the absence of field-device and energy measurements are identified to delimit the conclusions.

The rest of the article is structured as follows. Section 2 situates the work within plant disease classification, efficient CNNs, and attention-based refinements. Section 3 defines the dataset preparation, architecture, and transfer learning protocol. Section 4 reports classification, quantization, latency, interpretability, and limitation analyses. Section 5 closes with the main conclusions and remaining validation needs.

2. Related Work

Early plant disease vision systems usually followed a hand-engineered pipeline: isolate the leaf or lesion region, calculate color or texture descriptors, and train a conventional classifier. Reviews by Barbedo [5] and Zhang et al. [6] show that these approaches were

useful, but their behavior often depended on imaging assumptions and carefully selected features. Larger labeled image collections shifted the field toward CNN-based representation learning. Mohanty et al. [1] reported strong PlantVillage results, and later work by Sladojevic et al. [7] and Ferentinos [8] reinforced the value of deep models for plant pathology images. The remaining practical difficulty is that many accurate networks are too large or slow for the low-cost devices envisioned for field diagnosis.

Efficient CNN design addresses this tension by reducing operations, parameters, or memory traffic. MobileNet popularized depthwise separable convolution for mobile vision [9], and MobileNetV2 added inverted residual blocks with linear bottlenecks [3]. InceptionV3 [10], Xception [11], and EfficientNet [12] also target efficiency, but their storage and latency profiles differ markedly. For agricultural diagnosis, a model should therefore be judged by accuracy together with file size, memory demand, inference time, and conversion behavior. Khan et al. [13] examined plant disease recognition for edge devices and highlighted optimization and quantization. Duhan et al. [14] introduced RTR_Lite_MobileNetV2, and Li et al. [15] combined MobileNetV2 with coordinate attention, multi-branch convolution, and compression for tea-leaf disease identification. These studies show that MobileNetV2 variants are an active design family rather than a single fixed architecture.

Attention is one way to recover discriminative detail without wholesale enlargement of the backbone. Squeeze-and-excitation blocks recalibrate channels [16]; CBAM extends this idea by applying channel attention followed by spatial attention [4]. Such mechanisms are relevant to leaf imagery because class cues may be localized blemishes, discoloration, or texture changes rather than global object shape. Recent agricultural studies reflect this trend. Subburaj et al. [17] used a MobileNetV2 encoder in Plantention for multi-crop disease classification. Wen et al. [18] compared CA, ECA, SE, and CBAM within a lightweight tea-leaf disease and pest model. Pal et al. [19] combined lightweight modeling with Grad-CAM, Grad-CAM++, and LIME explanations. Together, these papers motivate attention-enhanced compact models while also showing that any proposed attention placement should be evaluated against a clear baseline.

Even with this progress, many reports still foreground classification scores more than deployment behavior. Recent reviews point to illumination changes, similar symptoms, overfitting, dataset bias, weak generalization, and implementation constraints as barriers to practical plant disease recognition [20–22]. Fewer studies report classification metrics alongside parameters, model storage, TensorFlow Lite size, and latency under a reproducible protocol. The present work addresses this narrower evidence gap by placing CBAM after the final MobileNetV2 map and before global average pooling, then measuring the incremental accuracy–cost effect relative to an unmodified MobileNetV2 baseline. It does not infer suitability for a specific device without target-hardware testing.

Table 1 positions the present study against representative work published in 2024–2025. The comparison is qualitative because the studies use different datasets, splits, preprocessing, hardware, and reporting conventions; their accuracy and latency values should not be treated as a controlled leaderboard. Importantly, recent work already combines MobileNetV2 with attention or other lightweight refinements. The distinction of the present study is therefore the evaluated single late-CBAM placement and its measured incremental trade-off against an unmodified MobileNetV2 baseline, not the invention of an attention-enhanced MobileNet family.

Table 1. Scope comparison with representative recent lightweight plant disease studies.

Study	Lightweight Strategy	Scope and Relation
Li et al. (2024) [15]	MobileNetV2, coordinate attention, multi-branch convolution, and compression	Tea-leaf evaluation; a broader modification using a different attention mechanism.
Duhan et al. (2025) [14]	RTR_Lite_MobileNetV2	Confirms that lightweight MobileNetV2 variants form an active, non-unique design family.
Subburaj et al. (2025) [17]	Attention model with a MobileNetV2 encoder	Multi-crop scope; this study instead isolates one late CBAM block.
Wen et al. (2025) [18]	Lightweight CNN comparing CA, ECA, SE, and CBAM	Provides an attention ablation that remains necessary here under one common protocol.
Pal et al. (2025) [19]	MobileNetV2 plus residual learning (3.51 M parameters)	Two datasets and Grad-CAM/Grad-CAM++/LIME; 99.47% on PlantVillage, under a different protocol.
This study	One CBAM block after the final MobileNetV2 convolutional map	Reports an incremental accuracy–cost trade-off; device and interpretability evidence remain outstanding.

3. Proposed Method

This section defines the dataset handling, network layout, and training protocol used for the late-CBAM MobileNetV2 evaluation. The central question is whether one attention block, placed near the classifier rather than throughout the backbone, can improve MobileNetV2 while keeping the additional storage and computation small. Claims about mobile or edge use are therefore framed as deployment motivation, with actual suitability reserved for device-specific validation.

3.1. Dataset Preparation

The experiments use 54,306 PlantVillage images in 38 classes. Each class was partitioned independently with `train_test_split`—70% for training and 30% temporary data—followed by an equal division of the temporary data into validation and test sets. Both operations used `random_state = 42`. The resulting directories contain 37,998 training images (69.97%), 8146 validation images (15.00%), and 8162 test images (15.03%). Because the archived training notebooks additionally specified `validation_split = 0.2`, the fitted generators used 30,416 images from the training directory and 1614 images from the validation directory; the test generator used all 8162 test images. This legacy configuration is reported explicitly rather than being described as an 80/10/10 split.

Training images were rescaled to $[0, 1]$ and augmented with rotations up to 25°, horizontal and vertical shifts of 0.2, zoom of 0.2, shear of 0.15, brightness factors from 0.8 to 1.2, horizontal flipping, and nearest-neighbor filling. Test images were only rescaled. MobileNetV2, MobileNetV2 + CBAM, and Xception used 224×224 inputs; InceptionV3 used its trained 299×299 input. A SHA-256 audit found no repeated filenames but identified 10 exact-image duplicate groups crossing directory splits; seven test images duplicated an image in training or validation. The audit detects byte-identical images only; perceptual near-duplicates or visually similar captures were not systematically clustered and remain a dataset-quality limitation. A sensitivity analysis therefore reports results both on all 8162 test images and after excluding the seven exact test duplicates. The controlled acquisition conditions of PlantVillage—typically isolated leaves and uniform backgrounds—remain a limitation for field generalization.

3.2. Overview of the Proposed Framework

The architecture is a transfer learning classifier. After preprocessing, an ImageNet-initialized MobileNetV2 backbone generates convolutional maps; CBAM then reweights these maps along channel and spatial dimensions; finally, global average pooling and a softmax classifier convert the refined representation into class probabilities.

The processing chain is written as

$$X \rightarrow F_{\text{MobileNetV2}}(X) \rightarrow F_{\text{CBAM}} \rightarrow F_{\text{GAP}} \rightarrow \hat{y}, \tag{1}$$

where X is the input image, $F_{\text{MobileNetV2}}(X)$ is the backbone output, F_{CBAM} is the representation after attention weighting, F_{GAP} denotes global average pooling, and $\hat{y}$ is the predicted probability vector. Figure 1 summarizes this processing chain.

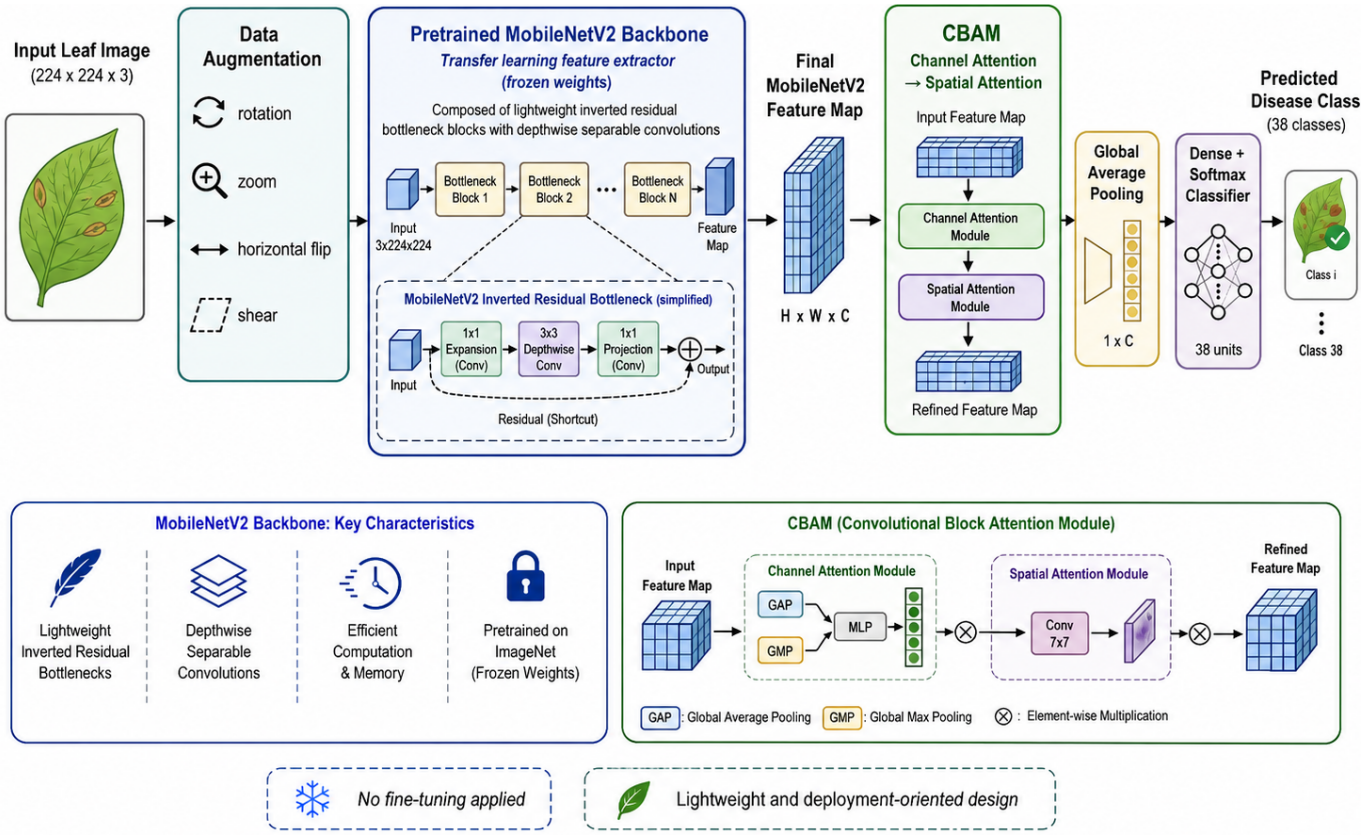

Figure 1. Overall late-attention MobileNetV2 architecture. A single CBAM block is applied to the final MobileNetV2 convolutional map before global average pooling and classification.

3.3. MobileNetV2 Backbone

MobileNetV2 was selected because its inverted residual and linear-bottleneck design provides a well-established, readily convertible TensorFlow implementation with substantially fewer parameters than the heavier InceptionV3 and Xception comparators. This choice defines the present scope; it does not establish superiority over MobileNetV1, MobileNetV3, MobileNetV4, EfficientNet-B0, GhostNet, or ShuffleNetV2, which require a common-protocol comparison.

As a baseline feature extractor, MobileNetV2 offers a compact TensorFlow/Keras implementation and a design intended for efficient vision workloads. Instead of relying primarily on full standard convolutions, it uses depthwise separable convolutions to factor spatial filtering from channel mixing. A conventional convolution learns filters over

spatial and channel axes together, whereas the separable form first filters each channel independently and then mixes channels with a 1×1 pointwise operation.

Given an input feature map, a standard convolution operation can be expressed as

$$Y = K * X, \tag{2}$$

where X denotes the input map, K the kernel, $*$ the convolution operator, and Y the resulting output.

For the depthwise part, each channel receives its own spatial kernel,

$$Y_d^c = K_d^c * X^c, \tag{3}$$

where X^c is channel c, K_d^c is that channel's depthwise kernel, and Y_d^c is the corresponding response. The pointwise stage then mixes the depthwise outputs

$$Y = K_p * Y_d, \tag{4}$$

where K_p is the pointwise kernel and Y_d collects the depthwise responses.

MobileNetV2 adds inverted residual blocks and linear bottlenecks to this separable-convolution design [3]. Each block expands the representation, applies depthwise filtering, and projects back to a lower-dimensional output; the linear bottleneck avoids applying a non-linearity at the narrow projection where information loss is more likely. These choices make MobileNetV2 a reasonable compact comparator against larger CNNs.

In this study, MobileNetV2 pretrained on ImageNet was used as the feature extraction backbone. The pretrained weights provide transferable low- and mid-level representations such as edges and textures. The backbone was initially frozen and its final 30 layers were then made trainable, providing limited task-specific adaptation while retaining most pretrained features.

3.4. Convolutional Block Attention Module

CBAM was selected because it combines channel selection ("what" features) with spatial selection ("where" features) in one sequential block. SE and ECA provide channel attention, whereas coordinate attention retains positional information through a different mechanism. The present experiment tests CBAM's two-dimensional refinement hypothesis; it does not claim superiority over those alternatives without a controlled attention-module ablation.

The motivation for adding attention is that a small backbone may under-emphasize local disease cues, especially when symptoms are small spots, color shifts, or texture disruptions. Attention weighting can bias the learned representation toward channels and locations that carry more class evidence.

In the proposed network, CBAM refines the MobileNetV2 output through two ordered masks [4]. The channel branch estimates which feature maps should be strengthened or weakened, and the spatial branch estimates where the informative responses lie within the feature map.

For an intermediate tensor $F \in \mathbb{R}^{H \times W \times C}$, the two CBAM stages are

$$F' = M_c(F) \otimes F, \tag{5}$$

$$F'' = M_s(F') \otimes F', \tag{6}$$

where $M_c(F)$ and $M_s(F')$ are the channel and spatial masks, respectively, and $\otimes$ indicates element-wise multiplication.

3.4.1. Channel Attention

The channel branch forms two descriptors by pooling each channel over its spatial extent. Average pooling captures the overall response level, while max pooling captures the strongest activation:

$$F^c_{\text{avg}} = \text{AvgPool}(F), \tag{7}$$

$$F^c_{\text{max}} = \text{MaxPool}(F). \tag{8}$$

Both descriptors pass through the same multilayer perceptron (MLP). Their outputs are added and squashed by a sigmoid,

$$M_c(F) = \sigma\Big(\text{MLP}(F^c_{\text{avg}}) + \text{MLP}(F^c_{\text{max}})\Big), \tag{9}$$

where σ is the sigmoid function. Multiplying this mask by F increases channels estimated to be useful and attenuates channels estimated to be less useful.

3.4.2. Spatial Attention

After channel weighting, the spatial branch estimates the location of informative responses. Average and maximum projections over the channel axis produce two two-dimensional descriptors:

$$F^s_{\text{avg}} = \text{AvgPool}_c(F'), \tag{10}$$

$$F^s_{\text{max}} = \text{MaxPool}_c(F'). \tag{11}$$

The descriptors are concatenated, filtered with a 7×7 convolution, and passed through a sigmoid,

$$M_s(F') = \sigma\Big(f^{7\times7}\Big([F^s_{\text{avg}}; F^s_{\text{max}}]\Big)\Big), \tag{12}$$

where $f^{7\times7}$ is the convolution and $[\cdot;\cdot]$ denotes concatenation. Multiplication with F' gives the final CBAM-refined tensor. Figure 2 summarizes the CBAM structure used in this study.

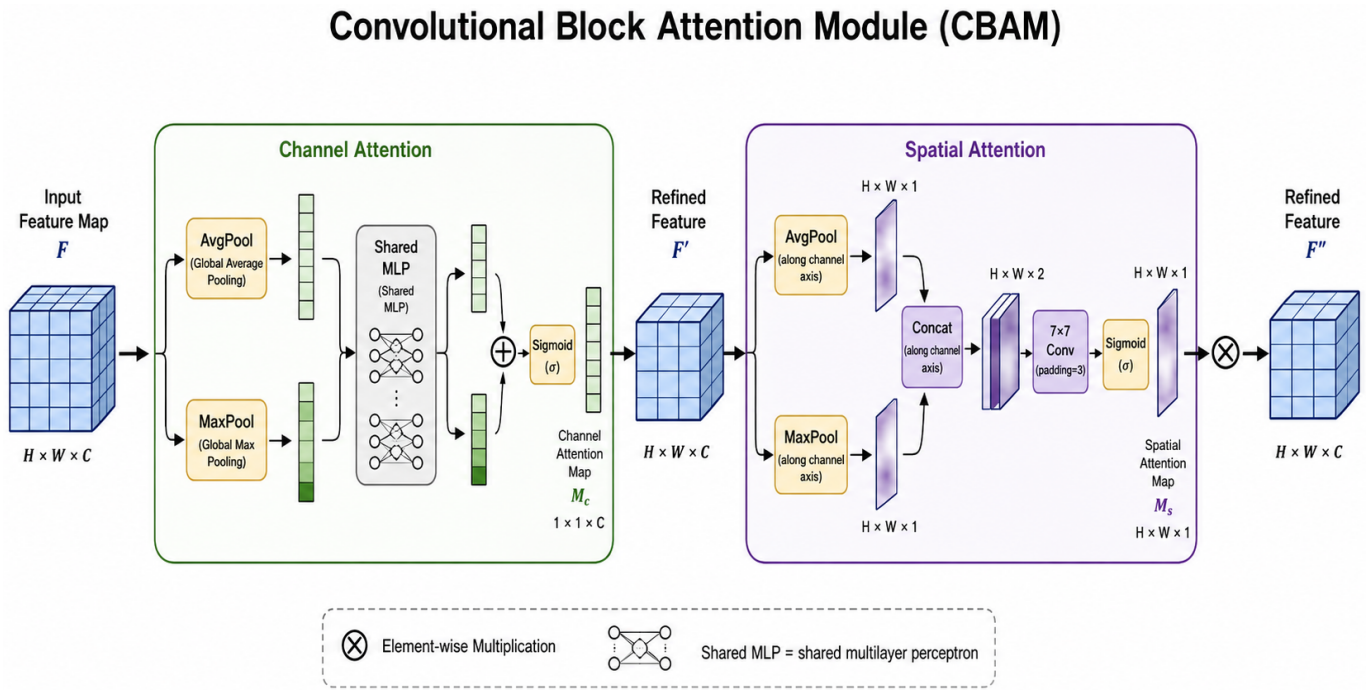

Figure 2. CBAM structure used in this study. Channel attention is applied first and spatial attention second, producing a refined version of the MobileNetV2 feature tensor.

3.5. Integration of CBAM with MobileNetV2

The late placement applies attention once to the most semantically developed convolutional map, avoiding repeated attention blocks throughout the backbone. Compared with early- or middle-stage attention, this placement acts after MobileNetV2 has compressed the representation to a small spatial map, which limits the extra operations and serialized storage introduced by CBAM. Compared with post-pooling attention, it preserves the spatial mask branch because the feature tensor still has two-dimensional structure before global average pooling. These are design motivations, not proof of optimality: early-stage, middle-stage, multi-block, and post-GAP attention placements require a dedicated ablation study.

In this configuration, the attention block is not repeated inside the backbone. MobileNetV2 first produces its final compact feature tensor, and CBAM then reweights that tensor immediately before the classification head. This choice keeps the intervention small and makes the additional cost easy to compare with the baseline.

Let F_m be the MobileNetV2 output:

$$F_m = F_{\text{MobileNetV2}}(X). \tag{13}$$

CBAM then computes

$$F_a = \text{CBAM}(F_m), \tag{14}$$

where F_a is the attention-refined tensor. Global average pooling produces

$$z = \text{GAP}(F_a). \tag{15}$$

The final dense layer applies softmax,

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \tag{16}$$

where $\hat{y}_i$ is the probability assigned to class i and $K = 38$ for this dataset.

The design therefore tests whether a targeted attention step can improve the MobileNetV2 representation without the storage and operation count of a much heavier backbone.

3.6. Classification Head

The classification head maps the refined tensor to the disease labels. Global average pooling collapses the spatial dimensions, and the dense softmax layer converts the pooled vector into a 38-class probability distribution.

Training minimizes categorical cross-entropy,

$$\mathcal{L} = -\sum_{i=1}^{K} y_i \log(\hat{y}_i), \tag{17}$$

where y_i is the one-hot target and $\hat{y}_i$ is the predicted probability for class i. This loss matches the mutually exclusive multi-class setting.

3.7. Transfer Learning Strategy

All four networks used transfer learning from ImageNet initialization. The approach reuses generic low- and mid-level visual filters while allowing the later layers and task-specific heads to adapt to PlantVillage disease classes.

For each backbone, all layers were first frozen and the final 30 layers were then unfrozen. The attention block and classification head were also trainable. This common

partial-fine-tuning rule allowed deeper features to adapt to disease texture and morphology while limiting optimization cost. It also corrects the earlier description of the backbones as completely frozen. The present study does not compare different unfreezing depths; such an ablation remains necessary to establish whether 30 layers provide the best accuracy–cost trade-off.

3.8. Proposed Model Summary

The evaluated MobileNetV2 + CBAM model is intended as a compact accuracy–cost compromise. MobileNetV2 supplies the efficient feature extractor, and the single late CBAM block adds channel and spatial reweighting before classification. The experiment therefore isolates the marginal effect of this attention addition rather than proposing a new backbone family. Table 2 summarizes the proposed MobileNetV2 + CBAM architecture.

The processing stages are:

1. Resize input leaf images to 224 × 224 pixels for the MobileNetV2-based models.
2. Apply training-time augmentation to expose the model to geometric and photometric variation.
3. Use ImageNet-pretrained MobileNetV2 to compute convolutional feature maps.
4. Reweight the final feature maps with CBAM channel and spatial masks.
5. Pool the refined tensor into a vector using global average pooling.
6. Predict one of the 38 classes with a softmax classifier.

Table 2. Summary of the proposed MobileNetV2 + CBAM architecture.

Stage	Component	Description
Input	Leaf image	Resized to 224 × 224 × 3.
Feature extraction	MobileNetV2 backbone	ImageNet-pretrained lightweight CNN with the final 30 layers unfrozen.
Attention refinement	CBAM	Sequential channel and spatial attention applied to the final MobileNetV2 feature maps.
Feature aggregation	Global Average Pooling	Converts the refined feature maps into a compact feature vector.
Classification	Dense + Softmax	Predicts one of 38 healthy or diseased plant leaf classes.

4. Experimental Results

4.1. Experimental Setup

Dataset partitioning, resizing, and augmentation are consolidated in Section 3.1. The held-out test set was used for the final classification results reported below.

The comparison includes InceptionV3, MobileNetV2, MobileNetV2 + CBAM, and Xception. MobileNetV2 is the direct lightweight baseline, the CBAM variant tests the proposed late-attention modification, and InceptionV3/Xception provide heavier reference architectures for interpreting the accuracy–cost trade-off.

All models were initialized with ImageNet weights. The final 30 backbone layers and the task-specific head were trainable; the earlier layers remained frozen. Training used at most 30 epochs, Adam with a learning rate of 1×10^{-5}, a batch size of 32, categorical cross-entropy, EarlyStopping, ReduceLROnPlateau, and ModelCheckpoint. The classification heads used global average pooling, dropout (0.4), a 1024-unit dense layer with $L_2 = 5 \times 10^{-4}$, a second dropout layer, and a 38-class softmax output.

Training was performed on a Windows 11 Pro laptop with an Intel Core i7-11800H CPU (eight cores/16 logical processors), 16 GB DDR4 RAM, and an NVIDIA GeForce RTX 3050 Ti Laptop GPU with 4 GB VRAM. The archived implementation used Python v3.10,

TensorFlow v2.13, and Keras. Deployment reanalysis used TensorFlow Lite v2.18 with the XNNPACK CPU delegate on the same Intel processor. Latency was measured with eight CPU threads, batch size one, 50 warm-up invocations, and 200 timed invocations; image decoding and resizing were excluded. Median and 95th-percentile latency are reported because isolated means are sensitive to operating-system scheduling. These laptop-CPU measurements are reproducible platform-specific benchmarks, not smartphone or microcontroller measurements.

Accuracy is reported over the full test set. Precision, recall, and F1-score use support-weighted averaging, with macro values discussed where class imbalance could affect interpretation. AUC is calculated as a macro one-vs-rest score over all 38 classes. Deployment evidence consists of parameter count, FLOPs, approximate MACs, serialized file size, dynamic-range-quantized TensorFlow Lite accuracy, and CPU latency.

4.2. Results and Discussion

Table 3 reports a consistent re-evaluation of the saved H5 models using their trained input sizes and explicit averaging rules. The ranking is stable across weighted and macro summaries: Xception remains the most accurate comparator, while MobileNetV2 + CBAM gives a modest improvement over the unmodified MobileNetV2 baseline. The gain is small but directionally consistent, increasing accuracy by 0.39 percentage points and macro F1-score by 0.41 percentage points. InceptionV3 performs close to the MobileNet variants when evaluated at its trained 299×299 resolution, but its computation and storage costs are much larger than MobileNetV2.

Table 3. Re-evaluation of the saved floating-point models. Weighted F1 uses class support; macro precision/recall/F1 and AUC give each class equal weight.

Model	Accuracy	Weighted F1	Macro P	Macro R	Macro F1/AUC
InceptionV3	96.89%	96.88%	95.91%	95.74%	95.74/99.96%
MobileNetV2	96.78%	96.73%	96.44%	95.66%	95.89/99.97%
MobileNetV2 + CBAM	97.17%	97.15%	96.70%	96.15%	96.30/99.97%
Xception	98.30%	98.30%	97.87%	97.84%	97.84/99.98%

Per-class precision, recall, F1-score, confusion matrices, and prediction arrays are not reproduced as full manuscript tables because the dataset contains 38 classes; they are provided in the reproducibility package so that class-level errors can be inspected without expanding the main text.

Training histories were reviewed to contextualize the final test metrics. Figure 3 shows accuracy and loss trajectories for the four models, and Figure 4 shows the corresponding AUC trajectories. These plots are used to inspect convergence and overfitting patterns rather than to replace held-out test evaluation.

The training curves indicate that the evaluated models converged within the training period. Figure 5 provides qualitative evidence about feature localization. In the first example, CBAM shifts the strongest response toward a broader central leaf region and corrects the baseline prediction. In the second example, both models attend to visible scab regions. In the failure example, both concentrate on the damaged leaf apex but predict the wrong disease class. The visualization therefore supports lesion-sensitive behavior in selected cases while also showing that CBAM does not consistently resolve symptom ambiguity. Quantitative explainability scores such as insertion/deletion, localization overlap, and faithfulness metrics were not computed; these are retained as future validation rather than inferred from Grad-CAM examples alone.

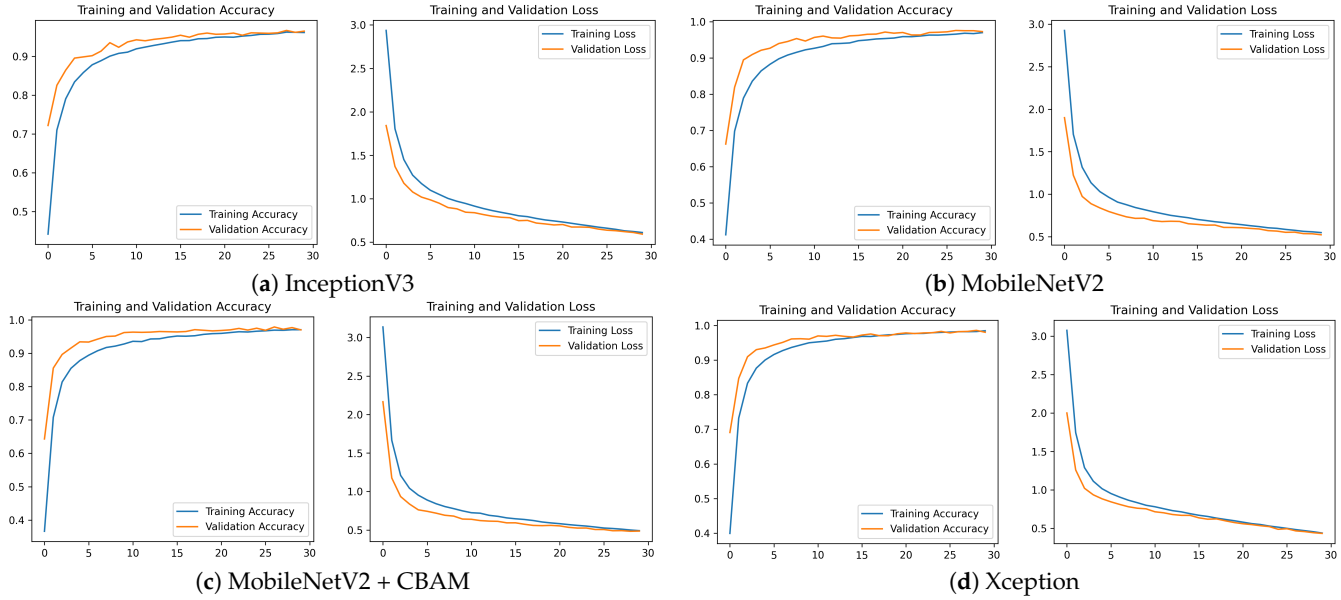

Figure 3. Training and validation accuracy/loss histories. (**a**) InceptionV3. (**b**) MobileNetV2. (**c**) MobileNetV2 + CBAM. (**d**) Xception.

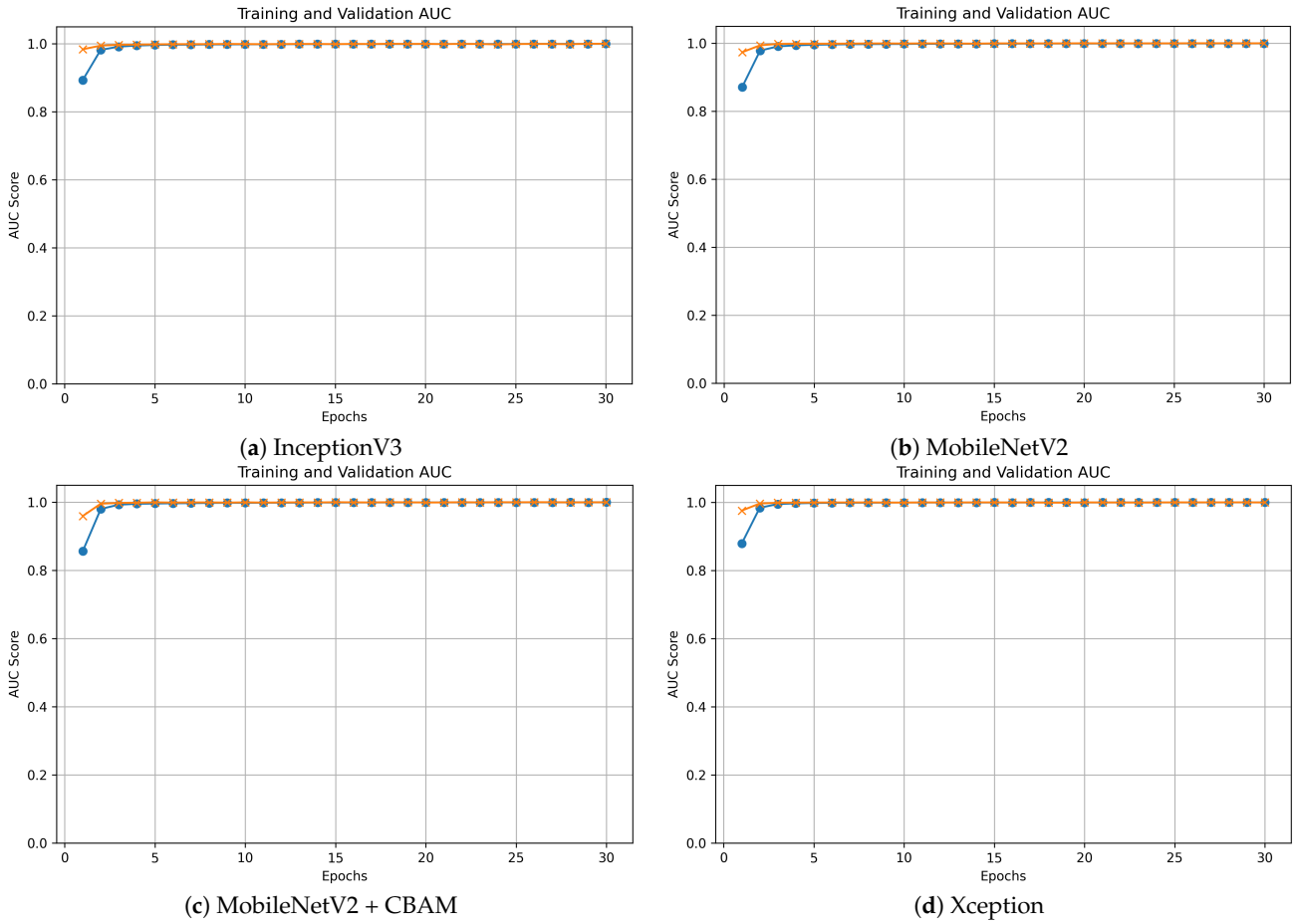

Figure 4. Training and validation AUC histories. Blue curves indicate training AUC, and orange curves indicate validation AUC. (**a**) InceptionV3. (**b**) MobileNetV2. (**c**) MobileNetV2 + CBAM. (**d**) Xception.

Figure 5. Matched Grad-CAM comparison for MobileNetV2 and MobileNetV2 + CBAM. Warmer heatmap colors indicate stronger Grad-CAM activation, and cooler colors indicate weaker activation. The first row shows a sample misclassified as healthy by MobileNetV2 but correctly classified as apple scab by the CBAM model. The second row is correctly classified by both models. The third row is misclassified by both models as tomato late blight, illustrating that attention localization does not guarantee class-specific discrimination.

Taken together, the single-training-run results show a small improvement over MobileNetV2. On the independently converted TensorFlow Lite models, CBAM improved paired test accuracy by 0.64 percentage points (95% CI: 0.31–0.96); it uniquely corrected 117 samples while MobileNetV2 uniquely corrected 65 (exact two-sided McNemar $p = 1.42 \times 10^{-4}$). This paired test indicates that the observed prediction difference is unlikely to arise from test-sample variation alone, but it does not replace mean $\pm$ standard deviation over independently trained seeds.

4.3. Deployment and Computational Analysis

Deployment analysis must consider more than test accuracy. Training-time logs first show that the lightweight models were also cheaper to fit under the same laptop-GPU

environment (Table 4). Table 5 then combines graph-level FLOPs, approximate MACs, serialized model sizes, and the standardized laptop-CPU latency benchmark.

Table 4. Training time measured from the archived notebook logs. Each model was trained for 30 epochs on the same Windows 11 laptop with an NVIDIA GeForce RTX 3050 Ti Laptop GPU.

Model	Total Training Time	Average Time/Epoch
InceptionV3	4 h 09 m 26 s	8.31 min
MobileNetV2	1 h 56 m 05 s	3.87 min
MobileNetV2 + CBAM	1 h 56 m 49 s	3.89 min
Xception	2 h 09 m 26 s	4.31 min

Table 5. Computational and CPU deployment comparison. Latency is TensorFlow Lite/XNNPACK, eight CPU threads, batch size one, 50 warm-up and 200 timed runs; preprocessing is excluded. TFLite sizes use MiB (2^{20} bytes).

Model	Params	GFLOPs	TFLite	Median	P95
InceptionV3	23.94 M	11.451	23.10 MiB	10.71 ms	12.27 ms
MobileNetV2	3.61 M	0.602	3.68 MiB	45.38 ms	106.96 ms
MobileNetV2 + CBAM	4.02 M	0.604	4.07 MiB	45.42 ms	102.54 ms
Xception	23.00 M	9.115	22.57 MiB	200.15 ms	304.13 ms

The central deployment pattern is that CBAM increases MobileNetV2's representational capacity more than its operation count, because the block is applied once to the final 7×7 feature map. Table 6 isolates this overhead. The median latency difference is below 0.1 ms in this XNNPACK benchmark, while the 95th-percentile values show operating-system scheduling variability and should not be interpreted as a stable speed advantage. Xception remains the most accurate model but is approximately 5.5 times larger than the CBAM TFLite file and requires about 15 times its approximate MACs.

Table 6. Incremental overhead of adding one late CBAM block to MobileNetV2. Size values use MiB (2^{20} bytes).

Metric	MobileNetV2	MobileNetV2 + CBAM	Increase
Parameters	3.61 M	4.02 M	+0.41 M (+11.35%)
GFLOPs	0.602	0.604	+0.0019 (+0.32%)
Approx. GMACs	0.301	0.302	+0.0010 (+0.32%)
H5 size	36.26 MiB	40.98 MiB	+4.72 MiB (+13.02%)
TFLite size	3.68 MiB	4.07 MiB	+0.40 MiB (+10.76%)
Median CPU latency	45.38 ms	45.42 ms	+0.05 ms (+0.10%)

The converter used `Optimize.DEFAULT` without a representative calibration dataset. This is dynamic-range weight quantization: model inputs and outputs remain float32 while eligible weights are quantized at runtime. Table 7 shows that conversion preserved most classification performance. The largest reduction was 0.71 percentage points for MobileNetV2, while MobileNetV2 + CBAM decreased by 0.47 points.

Table 7. Floating-point H5 and dynamic-range TFLite test performance. TFLite F1 is reported as both weighted and macro averaged; AUC is macro one-vs-rest.

Model	H5 Acc.	TFLite Acc.	ΔAcc.	TFLite F1 W/M	TFLite AUC
InceptionV3	96.89%	96.80%	−0.09 pp	96.80/95.58%	99.96%
MobileNetV2	96.78%	96.07%	−0.71 pp	96.07/95.42%	99.96%
MobileNetV2 + CBAM	97.17%	96.70%	−0.47 pp	96.69/95.89%	99.96%
Xception	98.30%	98.22%	−0.07 pp	98.23/97.75%	99.98%

Excluding the seven test images with an exact duplicate in another split changed TFLite accuracy by less than 0.01 percentage points for every model (e.g., 96.70% to 96.70% for MobileNetV2 + CBAM after rounding). Thus, the detected exact duplicates do not explain the reported ranking, although future split generation should group duplicate content before partitioning.

Overall, MobileNetV2 + CBAM is compact and slightly more accurate than MobileNetV2, with negligible additional graph operations in this late-placement design. The gain remains modest, plausibly because PlantVillage's uniform backgrounds and salient symptoms leave limited irrelevant content for spatial attention to suppress. The result is therefore best interpreted as an accuracy–cost trade-off observed on the measured dataset and hardware protocol, not as a general claim of deployment readiness. Energy consumption was not measured, and a laptop CPU is not a substitute for a target phone, Raspberry Pi, or microcontroller.

4.4. Limitations and External Validity

The study evaluates controlled-background image classification. PlantVillage does not capture field clutter, illumination changes, partial leaves, occlusion, overlapping foliage, camera motion, or heterogeneous symptom severity. Spatial attention may learn acquisition regularities as well as lesions, so the results should not be generalized to field deployment without external validation. The archived pipeline also used only a subset of the nominal training and validation directories and contained 10 exact cross-split duplicate groups; these details have been audited and disclosed, and a duplicate-excluded sensitivity analysis was added. Perceptual near-duplicates were not systematically clustered. Finally, each architecture represents one training run. The paired McNemar result and confidence interval quantify test-sample uncertainty for the saved models, but not seed-to-seed training variability. Independent reruns, early/middle/multiple/post-GAP attention-placement ablations, alternative attention modules, quantitative explainability metrics, and target-device memory, latency, and energy measurements remain necessary.

5. Conclusions

This study evaluated a late-attention MobileNetV2 classifier for PlantVillage disease recognition under explicit accuracy, size, quantization, training-time, and latency measurements. The architecture adds one CBAM block after the final MobileNetV2 convolutional map and before global average pooling, so the attention mechanism reweights the most semantically developed feature tensor without being repeated throughout the backbone. This placement is motivated by low additional computation and retention of a spatial mask before pooling, but it should be understood as a tested design choice rather than a proven optimal attention location.

Using 54,306 images from 38 PlantVillage classes, the saved-model re-evaluation produced 97.17% accuracy and 97.15% weighted F1-score for MobileNetV2 + CBAM, compared with 96.78% and 96.73% for MobileNetV2. Xception achieved the highest accuracy at 98.30%. Matched Grad-CAM examples indicated that CBAM can broaden lesion-centered activation and correct a baseline error, although one shared failure showed that attention localization alone does not guarantee the correct disease label.

The CBAM variant contains 4.02 million parameters, requires 0.604 GFLOPs (approximately 0.302 GMACs), and converts to a 4.07 MiB dynamic-range-quantized TFLite file. Its quantized accuracy was 96.70%, representing a 0.47-percentage-point decrease from the H5 evaluation. On an Intel i7-11800H CPU with TFLite/XNNPACK and eight threads, median batch-one latency was 45.42 ms (P95: 102.54 ms), nearly identical to MobileNetV2 in the

same benchmark. These results demonstrate compactness on the measured platform rather than universal real-time readiness.

The findings are constrained by the controlled-background dataset, legacy generator subsampling, 10 audited exact duplicate groups across splits, the absence of perceptual near-duplicate clustering, and the lack of independently trained seeds. Removing the seven exact duplicated test images did not materially change the rounded results, but future dataset construction should group duplicate content before partitioning. The study also leaves field clutter, illumination variation, occlusion, symptom severity, and domain shift unresolved. No phone, Raspberry Pi, microcontroller, or energy measurement was performed.

Future work should therefore prioritize:

- Repeated training runs with mean $\pm$ standard deviation reporting;
- Early-, middle-, multi-block, and post-GAP CBAM placement ablations;
- Comparisons with SE, ECA, coordinate attention, MobileNetV3/V4, EfficientNet-lite, GhostNet, and ShuffleNetV2 under one protocol;
- Quantitative explainability tests such as insertion/deletion, localization, and faithfulness metrics;
- Validation on field datasets with clutter, illumination variation, occlusion, and heterogeneous symptom severity;
- Latency, peak memory, and energy measurements on representative phones, Raspberry Pi-class devices, and microcontroller-class platforms where applicable.

Author Contributions: Conceptualization, E.U., M.A.A.M. and M.A.; methodology, E.U., M.A.A.M. and M.A.; software, E.U.; validation, E.U., M.A.A.M. and M.A.; formal analysis, E.U.; investigation, E.U.; resources, E.U.; data curation, E.U.; writing—original draft preparation, E.U.; writing—review and editing, E.U., M.A.A.M. and M.A.; visualization, E.U.; supervision, M.A.A.M. and M.A.; project administration, M.A.A.M. and M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The PlantVillage dataset is publicly available from its original source. The reproducibility package for this manuscript, including split scripts, training notebooks, saved H5 and TensorFlow Lite models, checksum files, evaluation scripts, prediction arrays, per-class reports, confusion matrices, and derived analysis outputs, is archived on Zenodo at https://doi.org/10.5281/zenodo.21464287, accessed on 22 July 2026. The archived scripts are intended to reproduce the reported saved-model, TFLite, duplicate-sensitivity, latency, and significance analyses from the raw PlantVillage directory structure and saved models.

Acknowledgments: The authors acknowledge the Department of Computer Science at Bishop's University for academic support during this research.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AUC	Area Under the Curve
CBAM	Convolutional Block Attention Module
CNN	Convolutional Neural Network
GAP	Global Average Pooling
GPU	Graphics Processing Unit
MLP	Multilayer Perceptron
TFLite	TensorFlow Lite

References

1. Mohanty, S.P.; Hughes, D.P.; Salathé, M. Using Deep Learning for Image-Based Plant Disease Detection. *Front. Plant Sci.* **2016**, *7*, 1419. [CrossRef]
2. Hughes, D.P.; Salathé, M. An Open Access Repository of Images on Plant Health to Enable the Development of Mobile Disease Diagnostics. *arXiv* **2015**, arXiv:1511.08060.
3. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*; IEEE Computer Society: Los Alamitos, CA, USA, 2018; pp. 4510–4520. [CrossRef]
4. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. CBAM: Convolutional Block Attention Module. In *Proceedings of the European Conference on Computer Vision*; Springer: Cham, Switzerland, 2018; pp. 3–19.
5. Barbedo, J.G.A. Digital Image Processing Techniques for Detecting, Quantifying and Classifying Plant Diseases. *SpringerPlus* **2013**, *2*, 660. [CrossRef] [PubMed]
6. Zhang, J.; Huang, Y.; Pu, R.; Gonzalez-Moreno, P.; Yuan, L.; Wu, K.; Huang, W. Monitoring Plant Diseases and Pests through Remote Sensing Technology: A Review. *Comput. Electron. Agric.* **2019**, *165*, 104943. [CrossRef]
7. Sladojevic, S.; Arsenovic, M.; Anderla, A.; Culibrk, D.; Stefanovic, D. Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification. *Comput. Intell. Neurosci.* **2016**, *2016*, 3289801. [CrossRef] [PubMed]
8. Ferentinos, K.P. Deep Learning Models for Plant Disease Detection and Diagnosis. *Comput. Electron. Agric.* **2018**, *145*, 311–318. [CrossRef]
9. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.
10. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*; IEEE Computer Society: Los Alamitos, CA, USA, 2016; pp. 2818–2826. [CrossRef]
11. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*; IEEE Computer Society: Los Alamitos, CA, USA, 2017; pp. 1800–1807. [CrossRef]
12. Tan, M.; Le, Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
13. Khan, A.T.; Jensen, S.M.; Khan, A.R.; Li, S. Plant Disease Detection Model for Edge Computing Devices. *Front. Plant Sci.* **2023**, *14*, 1308528. [CrossRef] [PubMed]
14. Duhan, S.; Gulia, P.; Gill, N.S.; Narwal, E. RTR_Lite_MobileNetV2: A Lightweight and Efficient Model for Plant Disease Detection and Classification. *Curr. Plant Biol.* **2025**, *42*, 100459. [CrossRef]
15. Li, Z.; Li, Y.; Yan, C.; Yan, P.; Li, X.; Yu, M.; Wen, T.; Xie, B. Enhancing Tea Leaf Disease Identification with Lightweight MobileNetV2. *Comput. Mater. Contin.* **2024**, *80*, 679–694. [CrossRef]
16. Hu, J.; Shen, L.; Sun, G. Squeeze-and-Excitation Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*; IEEE Computer Society: Los Alamitos, CA, USA, 2018; pp. 7132–7141. [CrossRef]
17. Subburaj, B.; Rohan, M.; Ananthanarayanan, S.; Won, D. Plantention: A General-Purpose, Lightweight and Attention-Based Model for Multi-Crop Leaf Disease Classification. *Results Eng.* **2025**, *26*, 105075. [CrossRef]
18. Wen, X.; Liu, Q.; Tang, X.; Yu, F.; Chen, J. A Lightweight Convolutional Neural Network for Tea Leaf Disease and Pest Recognition. *Plant Methods* **2025**, *21*, 129. [CrossRef] [PubMed]
19. Pal, C.; Karmakar, S.; Mukherjee, I.; Chakrabarti, P.P. A Lightweight and Explainable CNN Model for Empowering Plant Disease Diagnosis. *Sci. Rep.* **2025**, *15*, 30720. [CrossRef] [PubMed]
20. Zhao, J.; Xu, L.; Ma, Z.; Li, J.; Wang, X.; Liu, Y.; Du, X. A Review of Plant Leaf Disease Identification by Deep Learning Algorithms. *Front. Plant Sci.* **2025**, *16*, 1637241. [CrossRef] [PubMed]
21. Nyawose, T.; Maswanganyi, R.C.; Khumalo, P. A Review on the Detection of Plant Disease Using Machine Learning and Deep Learning Approaches. *J. Imaging* **2025**, *11*, 326. [CrossRef] [PubMed]
22. Sajitha, P.; Andrushia, A.D.; Anand, N.; Naser, M.Z. A Review on Machine Learning and Deep Learning Image-Based Plant Disease Classification for Industrial Farming Systems. *J. Ind. Inf. Integr.* **2024**, *38*, 100572. [CrossRef]