

Article

Chroma-Sense 2.0: A Memory-Efficient Two-Stage Pipeline for Lightweight On-Device Plant Disease Segmentation and Classification

Kiran Kumar Kethineni ^{1,*} , Azalea Tang ¹, Saraju P. Mohanty ¹  and Elias Kougianos ² ¹ Department of Computer Science and Engineering, University of North Texas, Denton, TX 76207, USA; azaleatang@my.unt.edu (A.T.); smohanty@ieee.org (S.P.M.)² Department of Electrical Engineering, University of North Texas, Denton, TX 76207, USA; elias.kougianos@unt.edu

* Correspondence: kirankumarkethineni@my.unt.edu

Abstract

On-device plant disease perception must reconcile two competing demands: enough spatial detail to localise diseased tissue within a field image and a memory and compute budget small enough for microcontroller-class hardware. Single-network solutions that jointly learn a pixel-wise mask and a fine-grained disease label tend to oversubscribe both the Flash and activation SRAM of such devices. This paper presents Chroma-Sense 2.0, a two-stage lightweight pipeline that decouples the two subproblems and sizes each stage for its own budget. The two stages run sequentially on the same frame: Stage 1 is a per-channel convolutional classifier, derived from Chroma-Sense, that names the disease, and Stage 2 is a compact ESPNet segmenter that produces a binary diseased-versus-healthy mask localising it. Because the stages run one after the other rather than concurrently, the peak working memory of the pipeline is the maximum of the two stages rather than their sum. We evaluate the pipeline on the in-the-wild PlantSeg dataset using a curated 10-species, 34-class subset and a leakage-controlled protocol in which all training crops are derived from PlantSeg's official training images and all reported metrics are measured on a held-out test set of 5002 crops built from the official test images. The segmentation stage attains a mean foreground recall of 0.97 (mean foreground IoU of 0.49; 0.53 pooled over pixels), a deliberately recall-oriented operating point. Against Fast-SCNN, a small U-Net, LR-ASPP, and DeepLabV3+, ESPNet is the smallest-footprint model (140k parameters, 193 KB Int8 Flash) while retaining the highest foreground recall; the per-channel classifier reaches accuracies comparable to much larger ImageNet-pretrained backbones (MobileNetV3 and EfficientNet) using 10–13× fewer parameters. End to end, the coupled pipeline classifies the disease correctly on 87.8% of the test crops. An on-device profile on the OpenMV H7 and H7 Plus shows that the binding constraint at 256 × 256 is the segmenter's ≈4 MB contiguous activation arena, rather than parameter Flash: Even on the 32 MB-SDRAM H7 Plus, the usable interpreter heap is only about 4 MB, and the arena cannot be allocated as a single contiguous block from it, so on the tested firmware, the classifier runs on microcontrollers while the segmenter does not; the full pipeline instead fits the gigabyte-scale single-board-computer tier (for example, Raspberry Pi or NVIDIA Jetson Nano), and enabling the segmenter to run on microcontrollers remains the open gap.



Academic Editor: Chunping Li

Received: 2 July 2026

Revised: 1 August 2026

Accepted: 5 August 2026

Published: 7 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and

conditions of the [Creative Commons](https://creativecommons.org/licenses/by/4.0/)[Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: lightweight deep learning; memory-efficient inference; semantic segmentation; ESPNet; separable convolution; per-channel CNN; plant disease detection; on-device inference; PlantSeg

1. Introduction

Infectious plant diseases substantially reduce crop yields worldwide, and the growers most affected often operate in rural environments with intermittent or no network connectivity [1–3]. Smart-agriculture systems increasingly aim to close the perception loop on the same embedded device that captures the field image, eliminating cloud latency and network dependence. A system that both localises and identifies disease on a microcontroller or single-board computer can flag infected regions in real time and log infections without connectivity or subscription cost.

The critical bottleneck for such systems is memory. Agricultural edge hardware provides heap SRAM in the hundreds of kilobytes and Flash in the low megabytes, an order of magnitude below even compact encoder–decoder segmentation models. Two orthogonal lines of prior work have reduced the cost of edge CNNs. One reduces per-parameter inference costs through depthwise separable convolutions, inverted residuals, and squeeze-and-excitation modules [4–7]. The other exploits domain structure: Lesion textures such as spots, rings, mosaics, and powdery patches are visually simpler than general object categories, and shallow networks often match deep ones on plant–disease benchmarks [8,9]. Our earlier work, Chroma-Sense [10], pursued this second direction: it processes the R, G, and B channels of a pre-cropped leaf through a lightweight per-channel extractor, achieving a small Flash footprint while matching competitive accuracy.

Chroma-Sense [10] is, however, a pure classifier: It produces a disease label from a pre-cropped leaf image but there is no spatial mask. Many downstream tasks, such as infection mapping, region-level logging, and targeted treatment planning, first require an indication of where the diseased tissue is, that is, a segmentation mask. Attaching a full segmentation decoder to a classifier, or training a single multi-task network for both masks and labels, pushes the combined model past the memory envelope of the target hardware and defeats the purpose of the lightweight backbone.

This paper presents Chroma-Sense 2.0, a two-stage lightweight pipeline that resolves this tension by decoupling the two objectives and solving each with a purpose-built stage inside its own memory budget. The two stages run sequentially on each frame. Stage 1 is a per-channel convolutional classifier, updated from Chroma-Sense [10], that names the disease; Stage 2 is a compact ESPNet [11] that produces a binary diseased-versus-healthy mask by localising it. ESPNet is a published efficient segmentation network built from efficient spatial pyramid (ESP) modules, and we adopt it here rather than propose a new backbone. Because the two stages run one after the other rather than concurrently, the peak working memory of the pipeline is the *maximum* of the two stages, not their sum. We size, benchmark, and profile the pipeline for the edge layer of the three-layer IoT hierarchy shown in Figure 1, and we provide an empirical assessment of the pipeline’s executability on current microcontroller-class hardware.

The remainder of this paper is organised as follows. Section 2 states the problem, the proposed solution, and the contributions, and it separates new work from reused components. Section 3 surveys related work. Section 4 introduces the pipeline. Section 5 details the Stage 1 classifier, and Section 6 outlines the Stage 2 ESPNet segmenter. Section 7 describes the dataset’s construction and the experimental setup. Section 8 presents the memory, compute, and edge deployment profile. Section 9 reports segmentation, baseline, classification, and end-to-end results. Section 10 discusses the operating point and limitations, and Section 11 concludes this paper.

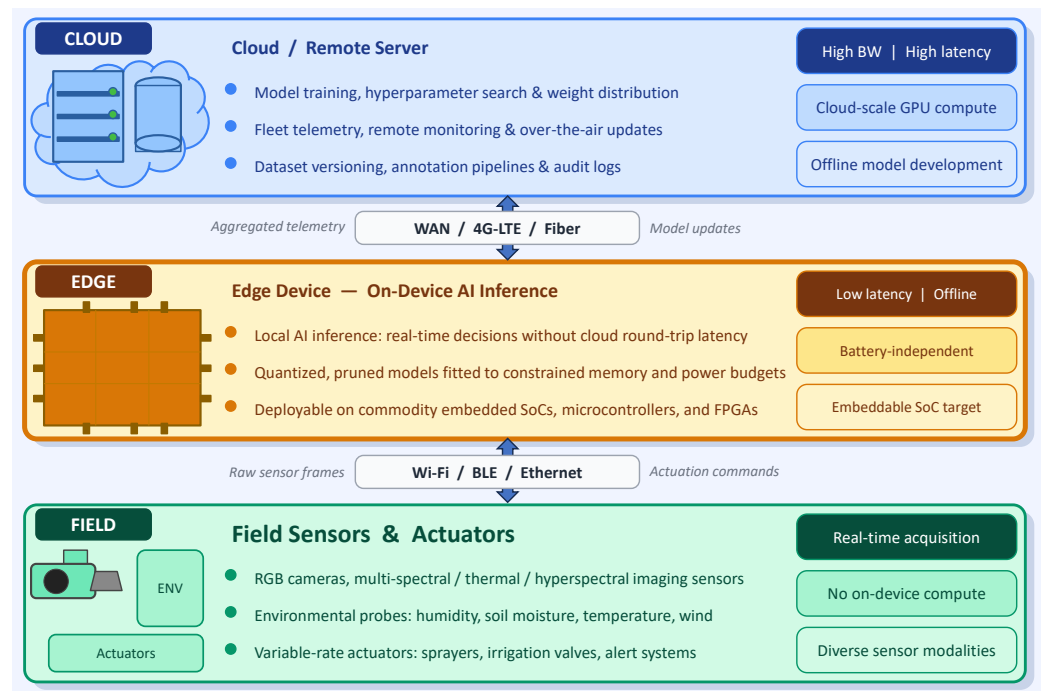


Figure 1. Three-layer IoT/edge deployment context; the proposed pipeline targets the edge layer.

2. Problem, Solution, and Contributions

2.1. The Problem

Deploying disease perception at the agricultural edge faces a compound problem.

Localisation and identification are different tasks with different information needs. Naming a disease is dominated by colour and texture signatures that discriminate one disease from another; localising diseased tissue mostly requires separating lesion tissue from healthy tissue and the background, independent of which disease is present. A classifier alone cannot say *where* the disease is, and a segmenter alone cannot say *which* disease it is.

Segmentation is the memory-limiting stage. Standard segmentation architectures that produce semantic masks, such as U-Net [12], DeepLabv3+ [13], and BiSeNet [14], require SRAM and Flash well above what agricultural microcontrollers provide. The decoder (transpose convolutions, skip-connection concatenation, and boundary-refinement layers) typically dominates the activation memory budget, and the largest feature maps in the network live inside it. A decoder that oversubscribes the activation arena cannot be deployed at all.

A single multi-task network does not reduce the memory footprint. Combining segmentation and classification into one network forces the shared encoder to be wide enough for both objectives while two heads compete for the same activation budget. The per-channel decomposition that makes classification memory-efficient is counter-productive for segmentation, which benefits from joint RGB cues; a decoder sized for masks discards the fine colour semantics the classifier needs. Decoupling the two objectives is therefore the memory-frugal choice.

2.2. The Solution

Chroma-Sense 2.0 decouples segmentation from classification and solves each with a stage sized for its subproblem.

Stage 1: Per-Channel Classification. Stage 1 is a compact per-channel CNN updated from Chroma-Sense [10] that processes the R, G, and B channels through independent extractors, fuses them, and emits a disease label for the frame.

Stage 2: Binary Segmentation with ESPNet. Collapsing every disease class into a single foreground label simplifies the learning problem, and the segmenter need only detect the presence of lesion texture. We use ESPNet [11], for which its ESP modules replace a heavy encoder with a point-wise reduction followed by a spatial pyramid of dilated convolutions fused by hierarchical addition and for which its light decoder uses two bilinear upsampling stages and a final resize rather than a symmetric transpose-convolution decoder. This yields the smallest parameter and Flash footprint among the lightweight segmenters that we benchmark (Section 9.2). The two stages run sequentially: first classification then segmentation on each frame.

2.3. Contributions and Scope of Novelty

We delineate the novel contributions of this work from its reused components as follows.

Reused Components. The Stage 2 backbone is ESPNet [11], an existing efficient segmentation architecture; the MobileNetV3-based baselines [5] we carry out comparisons against are also existing designs; the per-channel processing idea in Stage 1 originates with Chroma-Sense [10].

Novel Contributions. The contributions of this work are as follows:

1. An integrated, decoupled two-stage lightweight pipeline (per-channel classifier + ESPNet segmenter) that executes the two stages sequentially on the same frame, bounding peak working memory by the larger stage rather than by their sum. It is evaluated end to end as a coupled classify-then-segment system.
2. An updated per-channel classifier that replaces the original weight-shared 96×96 extractor of Chroma-Sense with three *independent* per-channel extractors operating on full-scene 256×256 inputs, pre-trained as a single 34-class representation and fine-tuned per species.
3. A Leakage-Controlled Benchmark Protocol on PlantSeg: fixed-size aspect-preserving crops (rather than whole-image stretching), source-grouped train/validation splitting, and a held-out test set constructed exclusively from PlantSeg's official test images.
4. Controlled Comparisons Against Lightweight Baselines for Both Stages Under Matched Protocols and Identical Int8 Quantisation: ESPNet against Fast-SCNN, a small U-Net, LR-ASPP, and DeepLabV3+ for segmentation and the per-channel classifier against ImageNet-pretrained MobileNetV3 and EfficientNet for classification on both accuracy and edge footprint.
5. An empirical on-device profiling and deployment assessment on OpenMV H7/H7 Plus, establishing that, at 256×256 , the binding constraint is the activation arena rather than parameter Flash and identifying which configurations are executable on the target hardware.

3. Related Work

3.1. Edge-Oriented CNN Backbones

MobileNet [4] introduced depthwise separable convolutions; MobileNetV2 [15] added inverted residuals and linear bottlenecks; MobileNetV3 [5] combined these with squeeze-and-excitation attention and neural architecture search. EfficientNet [16] and EfficientNetV2 [6] introduced compound scaling and training-aware searches. SqueezeNet [17] and ShuffleNet [7] minimised parameter counts, though its RAM profiles remain high relative to microcontroller budgets. MCUNet [18] co-designs the architecture and memory scheduler of IoT-class devices, demonstrating ImageNet [19]-quality classification within

a tight SRAM envelope. The common lesson is that depthwise-separable and pointwise operators survive the transition to microcontroller hardware, while large attention modules and concatenation-heavy decoders do not.

3.2. Plant Disease Classification

Early deep learning work applied full-width architectures, such as AlexNet and GoogLeNet, to curated leaf images [20], establishing expert-level accuracy on clean data. Archana et al. [21] applied a ResNet backbone, and Ferentinos [22] confirmed the finding across 58 plant–disease combinations. Ramcharan et al. [23] deployed a smartphone-hosted cassava classifier in the field. Multi-channel and capsule ensembles have also been tested [24]. More recent work targets the edge directly: Rakib et al. proposed a quantised few-layer CNN [25]; Bedi and Gole used a convolutional autoencoder for binary detection [26]; Chowdhury et al. [27] and Ashwinkumar et al. [28] brought EfficientNet- and MobileNet-class models to the multi-class problem. Li et al. [8] showed that very shallow networks match deep ones on plant–disease benchmarks, supporting the view that disease texture is low-dimensional. Chroma-Sense [10] exploited this structure by processing R, G, and B channels with a lightweight per-channel extractor. All of these operate on pre-cropped leaves and produce a label only; none localise the diseased region.

3.3. Plant Disease Segmentation and Efficient Decoders

Pixel-level plant–disease localisation has received less attention than classification. Barbedo [29] showed that lesion-level annotation improves model focus relative to image-level labels. U-Net [12] has been adapted to leaf-disease segmentation, delivering high IoU on server hardware but paying the full cost of its symmetric concatenation decoder, where each skip channel-wise concatenates encoder and decoder features and doubles the activation width at fusion. DeepLabv3+ [13] adds atrous spatial pyramid pooling and a refinement decoder. Feature pyramid networks [30] fuse multi-scale features by element-wise addition rather than concatenation, keeping the activation width constant. LR-ASPP [5] and BiSeNet [14] push toward the edge with narrow context branches and low-cost gates; Fast-SCNN [31] replaces the symmetric decoder with a lightweight feature-fusion module and a single upsampling operation; ESPNet [11] uses efficient spatial pyramid modules and a compact reduce–upsample–merge decoder. Object detectors such as Faster R-CNN [32] and YOLO [33] localise with bounding boxes rather than masks and do not fit sub-megabyte Flash. We adopt ESPNet as the Stage 2 segmenter and benchmark it directly against Fast-SCNN, a small U-Net, LR-ASPP, and DeepLabV3+ under a matched binary task in Section 9.2.

3.4. Positioning of the Proposed Work

Relative to the above, this paper (i) does not propose a new backbone family but packages existing efficient primitives into a decoupled two-stage pipeline under an edge memory budget; (ii) integrates a compact segmenter with an updated per-channel classifier and evaluates the two as a coupled system; and (iii) reports a leakage-controlled PlantSeg benchmark and an empirical on-device profile. Table 1 summarises our positioning.

Table 1. Positioning of the proposed pipeline relative to prior segmentation and edge-deployment work.

Work	Focus	Deployment	Seg.	Edge
Mohanty et al. [20]	Multi-class leaf classification (AlexNet/GoogLeNet).	Server class.	No	No
Ferentinos [22]	DNN plant-disease diagnosis, 58 classes.	Server class.	No	No
Ramcharan et al. [23]	Smartphone field classifier for cassava.	Mobile (smartphone).	No	Partial

Table 1. Cont.

Work	Focus	Deployment	Seg.	Edge
Faster R-CNN/YOLO [32,33]	Object detection + bounding-box localisation.	Server/GPU.	Bbox only	No
U-Net/DeepLab [12,13]	Accurate encoder–decoder pixel segmentation.	Server class.	Yes	No
MobileNet/MCUNet [4,18]	Generic edge-optimised backbones.	Edge/MCU.	No	Yes
Chroma-Sense [10]	Per-channel edge classifier for plant disease.	Edge (cropped leaf).	No	Yes
This work	Two-stage binary segmentation + classification pipeline.	Edge (resource constrained).	Yes	Yes

4. Proposed Two-Stage Pipeline

The pipeline is illustrated in Figure 2. A single RGB frame is processed by two stages in sequence: Stage 1, the per-channel classifier, names the disease present in the frame, and Stage 2, the ESPNet segmenter, produces a binary diseased-versus-healthy mask that localises it. The two models are independent—neither stage’s output is fed into the other—and they run one after the other on the same frame. This decoupling is what makes the pipeline realisable on a resource-constrained edge device: each stage is sized for its own subproblem, and because the stages execute sequentially rather than concurrently, only one stage’s activations occupy memory at a time.

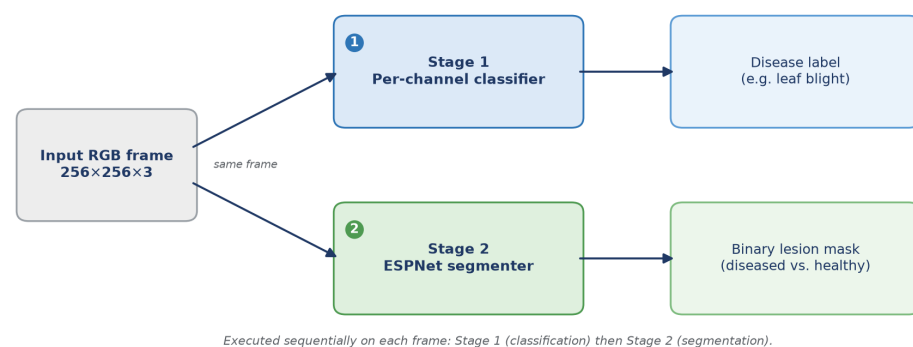


Figure 2. Two-stage edge pipeline: disease classification (Stage 1, per-channel CNN) followed by binary segmentation (Stage 2, ESPNet), run sequentially on the same input frame.

4.1. Why Decouple the Two Stages

Classification asks “which disease is this?” and is dominated by colour and texture signatures; segmentation asks “which pixels are diseased at all?” and mostly needs to separate lesions from healthy tissue and the background. Collapsing every disease class into a single foreground label makes the segmentation task *easier*, because the model no longer has to learn cross-disease distinctions on top of the foreground/background distinction. Conversely, a single multi-task network would require every parameter to serve two objectives simultaneously and would not reduce the activation budget. Decoupling lets the segmentation stage use joint RGB cues and a segmentation-appropriate decoder, and it lets the classification stage use the per-channel decomposition that makes it memory-efficient.

4.2. Sequential Execution on the Edge Device

On the device, the two stages are executed one after the other on each captured frame: The classifier (Stage 1) runs first and names the disease, and the segmenter (Stage 2) runs second on the same $256 \times 256 \times 3$ input and produces the mask. Because the two stages

never run concurrently, the peak activation memory of the pipeline is that of the larger stage alone (Section 8), not the sum of the two.

5. Stage 1: Per-Channel Classifier

Stage 1 names the disease. It retains the per-channel processing idea of Chroma-Sense [10], the observation that many lesion signatures are far more visible in one colour channel than the others, but it updates the architecture in two ways relevant to this pipeline: It processes the *full-scene* 256×256 frame (matching the segmenter) rather than a pre-cropped 96×96 leaf, and it uses three *independent* per-channel extractors rather than a single weight-shared extractor. The complete model has 307,532 parameters. Table 2 summarises this process.

Table 2. Stage 1 per-channel classifier, layer by layer (input $256 \times 256 \times 3$; 307,532 parameters for a 4-class head). The extractor is instantiated three times with independent weights, once per colour channel; its output shapes are per channel.

Layer	Output Shape	Stride
Per-channel feature extractor (R, G, B; independent weights)		
Conv2D $3 \times 3/8$	$128 \times 128 \times 8$	2
SepConv $3 \times 3/24$	$64 \times 64 \times 24$	2
SepConv $3 \times 3/24$	$64 \times 64 \times 24$	1
SepConv $3 \times 3/48$	$32 \times 32 \times 48$	2
SepConv $3 \times 3/48$	$32 \times 32 \times 48$	1
SepConv $3 \times 3/64$	$16 \times 16 \times 64$	2
SepConv $3 \times 3/64$	$16 \times 16 \times 64$	1
SepConv $3 \times 3/128$	$8 \times 8 \times 128$	2
SepConv $3 \times 3/128$	$8 \times 8 \times 128$	1
SepConv $3 \times 3/128$	$4 \times 4 \times 128$	2
SepConv $3 \times 3/128$	$4 \times 4 \times 128$	1
Fusion head (on concatenated R, G, B features)		
Concatenate R, G, B	$4 \times 4 \times 384$	n/a
Conv2D $1 \times 1/128$	$4 \times 4 \times 128$	1
Conv2D $1 \times 1/64$	$4 \times 4 \times 64$	1
Conv2D $1 \times 1/64$	$4 \times 4 \times 64$	1
GlobalMaxPool	64	n/a
Dense/64 ($\times 2$)	64	n/a
Dense softmax (pred)	n classes	n/a

5.1. Independent Per-Channel Extractors

The input is split into its R, G, and B channels, each a $256 \times 256 \times 1$ tensor. Each channel is processed by its own extractor: a strided 3×3 convolution to eight channels followed by a stack of SeparableConv2D blocks (each a depthwise 3×3 convolution and a pointwise 1×1 projection with batch normalisation and ReLU) that progressively halve spatial resolution and grow width ($24 \rightarrow 24 \rightarrow 48 \rightarrow 48 \rightarrow 64 \rightarrow 64 \rightarrow 128 \rightarrow 128$), emitting a $4 \times 4 \times 128$ feature cube per channel. Using independent extractors, rather than the weight-shared extractor of the original Chroma-Sense, allows each colour channel to learn its own filters, which we found necessary for the harder in-the-wild PlantSeg imagery; the cost is a larger parameter count, which is still well within the edge Flash budget (Section 8).

5.2. Fusion Head

The three $4 \times 4 \times 128$ cubes are concatenated to $4 \times 4 \times 384$ and mixed by three successive 1×1 convolutions ($128 \rightarrow 64 \rightarrow 64$), a pointwise fusion that recovers the cross-channel correlation deliberately sacrificed inside the per-channel extractors. The result is reduced by *global max pooling* rather than global average pooling: For sparse lesion activations, max pooling preserves the strongest local response instead of diluting it over the full spatial support, and it is robust to Int8 quantisation because the largest pre-quantisation value remains largest after quantisation. Two Dense(64) layers and a final softmax (pred) produce the per-species disease label.

6. Stage 2: ESPNet Segmentation Backbone

Stage 2 is ESPNet [11], a fully convolutional encoder–decoder that takes a $256 \times 256 \times 3$ RGB image and emits a $256 \times 256 \times 2$ softmax map (healthy background vs. diseased foreground). We use the published ESPNet design that is unchanged in structure and train it for the binary task; we do not modify the architecture. Table 3 summarises the network by stage.

Table 3. Stage 2 ESPNet by stage (input $256 \times 256 \times 3$; 140,423 parameters total).

Stage	Output Shape	Note
Stem Conv $3 \times 3/16$, s2	$128 \times 128 \times 16$	skip tap s_1
MaxPool + $2 \times$ ESP(64)	$64 \times 64 \times 60$	skip tap s_2
MaxPool + $3 \times$ ESP(128)	$32 \times 32 \times 125$	context
Conv $1 \times 1/128$	$32 \times 32 \times 128$	project
Up $\times 2$ + concat s_2 , $1 \times 1/64$	$64 \times 64 \times 64$	decode
Up $\times 2$ + concat s_1 , $1 \times 1/32$	$128 \times 128 \times 32$	decode
Resize + Conv $1 \times 1/2$ softmax	$256 \times 256 \times 2$	mask

6.1. Efficient Spatial Pyramid (ESP) Modules

ESPNet’s efficiency comes from the ESP module, which replaces a standard convolution with two cheaper steps. First, a point-wise (1×1) convolution *reduces* the channel width to $n = \lfloor f/5 \rfloor$. Second, the reduced tensor is passed through five parallel 3×3 convolutions with dilation rates $\{1, 2, 4, 8, 16\}$, forming a *spatial pyramid* that covers a large effective receptive field at low cost. To suppress the gridding artefacts that dilated convolutions can introduce, the pyramid outputs are combined by *hierarchical feature fusion*: Each dilated output is added to the accumulated sum of the lower dilation rates before the five branches are concatenated ($5n$ channels), batch-normalised, and activated. Because the expensive spatial convolutions operate on only n channels, an ESP module is markedly cheaper than a dense convolution of the same nominal width.

6.2. Encoder, Decoder, and Head

The encoder begins with a strided 3×3 convolution to $128 \times 128 \times 16$ and then alternates max-pooling with ESP modules at increasing widths (two ESP modules at 64×64 with 60-channel output and three at 32×32 with 125-channel output), followed by a 1×1 projection to 128 channels. The decoder is deliberately light: It bilinearly upsamples $2 \times$ and concatenates with the matching encoder tap (64×64), applies a 1×1 projection to 64 channels, upsamples $2 \times$ again, concatenates with the first encoder tap (128×128), projects to 32 channels, and finishes with a bilinear resize to 256×256 and a 1×1 convolution to the two-class logits. There is no symmetric transpose-convolution stack. The complete network has 140,423 parameters, an order of magnitude fewer than the

encoder–decoder baselines in Section 9.2, and its layer-wise activation budget is dominated by the pre-output feature map, as quantified in Section 8.

7. Dataset and Experimental Setup

7.1. Source Dataset

All experiments use the PlantSeg dataset [34,35], a collection of in-the-wild diseased-plant images with binary lesion masks. Unlike curated single-leaf collections, PlantSeg images are photographed in the field: They vary in size, lighting, background clutter, and plant overlap, which makes them a more realistic and more challenging evaluation corpus. From the broader dataset, we use a curated subset of 10 plant species spanning 34 disease classes (Table 4). The subset is simply the set of species we target for deployment; it is not selected to favour any model, and no per-species class is excluded. We do not use PlantVillage or any other supplementary corpus for the reported results.

Table 4. Curated PlantSeg subset and its data distribution: 10 species and 34 disease classes, with per-species training-crop counts (balanced to 400 per class) and held-out test-crop counts.

Plant	Classes	Disease Labels	Train Crops	Test Crops
Apple	4	black rot, mosaic virus, rust, scab	1600	600
Banana	4	anthracnose, black leaf streak, cigar end rot, cordana leaf spot	1600	600
Bell Pepper	4	bacterial spot, blossom end rot, frog eye leaf spot, powdery mildew	1600	533
Cabbage	3	alternaria leaf spot, black rot, downy mildew	1200	450
Celery	2	anthracnose, early blight	800	269
Cherry	2	leaf spot, powdery mildew	800	300
Corn	3	gray leaf spot, northern leaf blight, rust	1200	450
Peach	3	brown rot, leaf curl, scab	1200	450
Soybean	5	bacterial blight, brown spot, downy mildew, frog eye leaf spot, mosaic virus	2000	750
Tomato	4	late blight, leaf mold, mosaic virus, septoria leaf spot	1600	600
Total	34	10 species	13,600	5002

7.2. Fixed-Size, Aspect-Preserving Crops

The models operate at 256×256 . Resizing a whole field image to 256×256 would stretch or squash lesion features and distort the very textures on which the models rely. We therefore build all data from *fixed-size* 256×256 crops taken at the native spatial scale, rather than from stretched whole images. Figure 3 shows representative crops with their lesion masks: the crops preserve real lesion appearance, in-the-wild backgrounds, and aspect ratios.

Two crop sets are built. The training/validation set is cropped only from PlantSeg’s *official training images*: random 256×256 windows of the image and mask together (with light scale jitter), keeping a crop only if its mask contains more than 10% foreground so that a lesion is actually present. Each disease class is balanced to exactly 400 crops (Table 4), topping up scarce classes with augmentation (flips, rotation, brightness, contrast, colour, and sharpness; geometric transforms applied to image and mask and photometric transforms applied to the image only). This uniform per-class distribution yields 13,600 training crops; the held-out test set is left at its natural per-class counts and is never rebalanced. The test set is cropped only from PlantSeg’s *official test images*, with *no augmentation*, using a foreground rule relative to each source image’s own lesion density (a crop is kept if its foreground ratio is at least a fixed fraction of the source image’s ratio) so that sparse-lesion

images still yield valid crops. This yields a held-out test set of 5002 crops. The per-class test counts appear in Table 4.

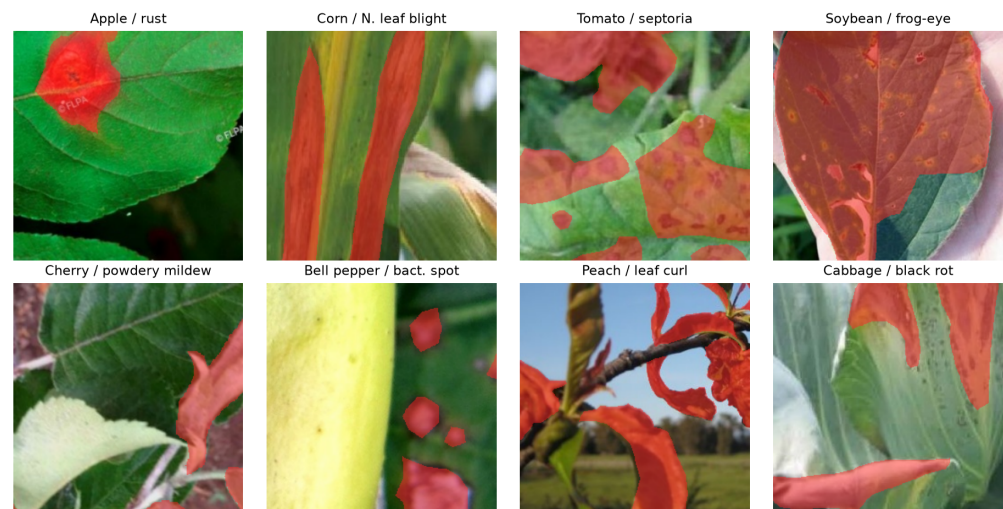


Figure 3. Representative 256×256 crops from the curated PlantSeg subset, with the binary lesion mask overlaid in red. Crops are taken at the native spatial scale (no whole-image stretching), preserving lesion appearance and in-the-wild background.

7.3. Train/Test Separation and Leakage Control

Leakage is controlled at three levels. (i) *Partition Firewall*: All training crops are derived from PlantSeg’s official training images, and all test crops are derived from their official test images, so no source image contributes to both. (ii) *Source-Grouped Validation*: The validation split is carved by the *source image*, so augmented crops of the same source never straddle the train/validation boundary. This addresses the per-crop leakage that a naive random split would introduce. (iii) *No Augmentation in Test*: The test set contains only real crops. All accuracy and IoU numbers reported in Section 9 are measured on the held-out test set.

7.4. Training Protocol

Stage 1 (Classification). A single 34-class classifier is first trained as a shared representation over the pooled PlantSeg crop set (Adam, 10^{-3}); then, for each species, a fresh head is attached, and all layers except the final pred softmax are imported from the pre-trained model by positional, shape-matched weight transfer; the head is warmed up with the body frozen for a few epochs, and the whole model is then fine-tuned (Adam, 5×10^{-4} , batch 32) with inverse-frequency class weighting, learning-rate reduction on plateau, and early stopping.

Stage 2 (Segmentation). Each species’ ESPNet is first pre-trained as a plant-agnostic binary lesion segmenter on the pooled PlantSeg crops (all species) for 30 epochs and then fine-tuned per species for up to 30 epochs on that species’ training crops. The loss is a foreground-weighted pixel-wise cross-entropy with a foreground weight of 20, which deliberately prioritises recall (not missing lesion pixels) over precision. Optimisation uses Adam (pre-train learning rate of 10^{-3} , fine-tune 5×10^{-4} , and batch of 16), with learning-rate reductions on plateau and early stopping on validation loss. Both stages use the same train/validation/test crops.

Reproducibility. All runs fix the random seed to 42 and use TensorFlow 2.20/Keras 3 (Python 3.12) on a single NVIDIA A100 GPU. Inputs are scaled to $[0, 1]$; training augmentation consists of random horizontal and vertical flips, 90° rotations, and ± 0.15 brightness jitter (geometric transforms applied to both image and mask, photometric to the image

only). The validation split is 10% of the training *sources*, carved by the source image. Both fine-tuning stages monitor validation loss with ReduceLROnPlateau (factor of 0.5, patience of 6, and minimum learning rate of 10^{-6}) and EarlyStopping (patience 12, best weights restored). The epoch budgets are as follows: Stage 1 shared pre-training: 150 epochs; Stage 1 head warm-up: 3 epochs, followed by per-species fine-tuning up to 40 epochs; Stage 2 pre-training: 30 epochs, followed by per-species fine-tuning up to 30 epochs. Full-integer Int8 conversion uses a representative subset of the training crops. The lightweight baselines (Sections 9.2 and 9.4) follow the same protocol.

7.5. Metrics

Segmentation is reported as foreground recall, precision, IoU, and Dice; the recall–IoU gap is the expected footprint of the recall-weighted loss. Classification is reported as accuracy and macro-averaged F1. The end-to-end system is scored by running the segmenter and classifier on every test crop and reporting both the classification accuracy and the segmentation quality, including on the *considered* subset (crops for which their disease is also named correctly), since a mask is most useful when the disease is identified.

8. Memory, Compute, and On-Device Deployment

This section quantifies the footprint of each stage and reports what actually runs on the target hardware. We target two OpenMV boards built on the same STM32H743 Cortex-M7 at 480 MHz: the base H7 (1 MB internal SRAM, 2 MB internal Flash) and the H7 Plus, which adds 32 MB of external SDRAM. Both run MicroPython v1.28 with the TensorFlow Lite for Microcontrollers (TFLite-Micro) interpreter.

8.1. What Determines the Footprint

Two quantities matter on-device, and they reside in different memories. *Weights* are constant and reside in Flash; the size of the full-integer Int8 .tflite file is the Flash demand. *Activations* are transient and reside in RAM: TFLite-Micro allocates a single contiguous *tensor arena* once at load time, sized to hold the tensors that are simultaneously live at the busiest point of the forward pass, and it reuses that one block for every layer. The RAM demand is therefore the size of that arena, not the sum of all activations. We estimate the arena as twice the largest single activation tensor, a ping-pong upper bound for a static planner; the exact value depends on the planner, but it is dominated by that largest tensor. Compute is reported as multiply–accumulate operations (MAdds), a hardware-independent count obtained analytically per layer.

8.2. Per-Stage and Pipeline Footprint

Table 5 lists the two stages and the pipeline. Weights are small for both: 193.1 KB for ESPNet and 458.8 KB for the classifier, which is 651.9 KB combined, a small fraction of either board’s Flash. The arenas, however, differ by an order of magnitude. The classifier downsamples to 4×4 almost immediately, so its largest tensor is the $256 \times 256 \times 3$ input (192 KB at Int8), and its arena is only 384 KB. ESPNet instead keeps a full-resolution feature map near its output: The $256 \times 256 \times 32$ pre-output tensor is 2 MB at Int8, which puts its arena between 2 MB (the tensor itself) and about 4 MB (the ping-pong upper bound). Because the two stages run sequentially (Section 4) and never hold activations at the same time, the pipeline’s arena is the *maximum* of the two stages, which is about 4 MB and set by the segmenter, whereas its Flash is the *sum* of 651.9 KB.

Table 5. Int8 footprint of each stage and the pipeline and whether each loads and runs on the two OpenMV boards. Flash is the full-integer .tflite size; the arena is the contiguous activation buffer, estimated as twice the largest activation tensor. For the pipeline row, Flash and MAdds are the per-frame sums of the two stages and the arena is their maximum since the stages run sequentially.

Model	Params	MAdds (M)	Flash (KB)	Arena (KB)	Runs on H7	Runs on H7 Plus
Stage 1: per-channel classifier	307,532	43.1	458.8	384	Yes	Yes
Stage 2: ESPNet segmenter	140,423	270.0	193.1	4096	No	No
Full pipeline	447,955	313.1	651.9	4096	No	No

8.3. The Binding Constraint Is the Contiguous Arena, Not Flash or Total SDRAM

The Flash demand (651.9 KB) sits well within both boards, so weight storage is never the limit. The limit is the tensor arena, specifically the largest *contiguous* free block from which it can be allocated.

On the base H7, the SRAM left for the interpreter after the accounting for the frame buffer and firmware is on the order of 512 KB. The classifier’s 384 KB arena fits, so it loads and runs. ESPNet cannot run at any buffer-reuse setting: its single largest activation, the 2 MB pre-output map, already exceeds the entire 512 KB heap roughly fourfold, so no arena small enough can exist.

H7 Plus adds 32 MB of SDRAM, which naively dwarfs the 4 MB arena. This SDRAM, however, is not a flat user heap. The OpenMV firmware statically partitions it, reserving large fixed regions for the frame buffer and firmware, so the MicroPython garbage-collected heap from which the TFLite arena must be carved is capped near 4 MB rather than at the full 32 MB. This is directly observable on the board (OpenMV 4 Plus, STM32H743, OpenMV firmware v5.0.0): At load, the interpreter reports about 4.1 MB free, and the ESPNet graph itself loads successfully (it accepts its $256 \times 256 \times 3$ input). The failure occurs at the next step, when the interpreter tries to allocate the activation arena. This arena must be a single contiguous buffer, and even at the optimistic lower end of its size—the 2 MB largest tensor itself, below our 4 MB ping-pong upper bound—it claims a large fraction of the ≈ 4 MB heap; once the interpreter’s own working memory, allocation overhead, and heap fragmentation are added, no contiguous block large enough remains and the tensor allocation fails, even though the reported free figure looks sufficient. The classifier’s 384 KB arena, in contrast, is a small fraction of the same heap and allocates with a wide margin. The binding quantity is therefore the largest *contiguous* block the heap can supply for the arena, not the total SDRAM and not the weight Flash: a model for which its arena is a large fraction of the free-heap size cannot be allocated even when the board nominally carries tens of megabytes of SDRAM.

8.4. On-Device Result

Consistent with the above, the classifier loads and runs on both H7 and H7 Plus, whereas ESPNet runs on neither at 256×256 . On H7, a single activation already exceeds the SRAM heap. On H7 Plus, the graph loads but the activation arena, for which its size approaches the ≈ 4 MB free heap, cannot be allocated, so inference never starts. Smaller modules with far less memory, such as Arduino Nicla Vision and Raspberry Pi Pico, cannot host the segmenter either. We do not report wall-clock latencies or frame rates because the segmenter does not run on the tested boards; any such figure would be an off-device estimate rather than a measurement, so we give the compute cost only as MAdds (Table 5).

8.5. Positioning on the Edge-Device Spectrum

“Edge” spans a wide hardware range, and the two stages land on different parts of it. At the microcontroller tier (OpenMV H7 and H7 Plus, Arduino Nicla Vision, Raspberry Pi Pico), usable memory is a few hundred kilobytes to a few megabytes and is the binding resource: The classifier, with its 384 KB arena, runs here today, whereas the segmenter, with its ≈ 4 MB arena at 256×256 , does not. At the single-board-computer and embedded-GPU tier (Raspberry Pi, NVIDIA Jetson Nano), a general-purpose operating system exposes gigabyte-scale RAM, and the Jetson additionally provides a GPU, so the full pipeline has ample memory headroom and is free of the arena constraint that blocks it on the microcontrollers. The two-stage design therefore degrades gracefully across the spectrum: the classification stage already reaches the smallest microcontrollers, while the full segment-and-classify pipeline targets the single-board-computer tier.

The gap this work makes precise is narrow and specific. Microcontroller-class edge hardware is limited not by compute or weight storage but by the size of the single contiguous activation arena it can allocate, and at 256×256 , the segmenter’s arena is several times that budget. Closing this gap so that full-resolution segmentation also reaches the microcontroller tier is the main deployment direction; candidate routes are a bare-metal runtime that places the arena directly in SDRAM (bypassing the interpreter heap cap), a segmenter with a smaller peak activation, or resolution-adaptive inference that trades spatial detail for arena size only when the hardware requires it. Lowering the fixed input resolution would also shrink the arena in proportion to the largest feature map but at the cost of spatial details that the segmentation task depends on, so we do not adopt this as the default.

8.6. Quantisation and Pruning

Both models are converted to full-integer Int8 (uint8 input and output) with TensorFlow Lite using a representative dataset; the Flash and arena figures above are for these deployed Int8 models. We also explored 20% magnitude pruning with fine-tuning: On apple, at 50% sparsity, the pruned ESPNet fine-tunes back to roughly 94% of its unpruned foreground IoU and the pruned classifier to roughly 96% of its unpruned accuracy, so most of the loss from pruning is recovered. A dense Int8 .tflite, however, does not shrink from magnitude pruning, because zeroed weights still occupy a byte, and the arena is unchanged; realising footprint savings would require structured (channel) pruning or a sparse-aware runtime. We therefore report pruning only as a fine-tunable starting point, not as a deployed size reduction.

9. Results

9.1. Segmentation Results

Table 6 reports per-species ESPNet (Stage 2) segmentation on the held-out test set. The mean foreground recall is 0.97 (pooled over pixels, 0.98), confirming that the recall-weighted loss reliably captures lesion tissues. The mean foreground IoU is 0.49 (pooled 0.53); the gap between recall and IoU is the expected footprint of the recall-oriented operating point, which over-covers lesion boundaries and thus lowers precision. IoU varies with lesion morphology: species with large, contiguous lesions (soybean 0.74, corn 0.61, peach 0.60) score well, while species with small or diffuse lesions (celery 0.17, cherry 0.37) score lower even though recall stays high.

9.2. Segmentation Baseline Comparison

To justify ESPNet as the Stage 2 backbone, we benchmark it against four lightweight segmentation baselines (Fast-SCNN [31], a small U-Net [12], and LR-ASPP and DeepLabV3+

with MobileNetV3 backbones [5,13]) under an identical protocol: the same binary foreground task, the same $256 \times 256 \rightarrow 256 \times 256$ input/output, the same recall-weighted loss and training schedule, the same per-species fine-tuning and held-out test set, and the same full-integer Int8 quantisation. Table 7 reports the mean over the 10 species alongside the edge footprint.

Table 6. Stage 2 ESPNet per-species segmentation on the held-out PlantSeg test set (FG = foreground).

Plant	FG Recall	FG Precision	FG IoU	FG Dice
Apple	0.968	0.564	0.553	0.712
Banana	0.983	0.543	0.538	0.700
Bell Pepper	0.914	0.446	0.428	0.599
Cabbage	0.991	0.434	0.432	0.604
Celery	0.969	0.170	0.169	0.289
Cherry	0.933	0.378	0.368	0.538
Corn	0.986	0.617	0.612	0.759
Peach	0.987	0.600	0.595	0.746
Soybean	0.994	0.745	0.742	0.852
Tomato	0.985	0.453	0.450	0.620
Mean	0.971	0.495	0.489	0.642
Pooled	0.980	0.541	0.535	0.697

Table 7. Lightweight binary-segmentation baselines under a matched protocol (mean over 10 species; footprint after full-integer Int8). All models share the same task, loss, schedule, splits, and quantisation.

Model	Params	Int8 (KB)	RAM (KB)	mIoU	mDice	mRecall
U-Net (small)	7,855,522	7755.5	12,288	0.481	0.635	0.965
Fast-SCNN	946,098	1189.1	16,384	0.508	0.658	0.966
LR-ASPP (MNV3)	1,055,010	1272.5	2080	0.595	0.735	0.955
DeepLabV3+ (MNV3)	3,816,186	3998.8	16,384	0.610	0.748	0.963
ESPNet (ours)	140,423	193.1	4096	0.489	0.642	0.971

Two observations follow from Table 7. First, ESPNet has by far the smallest static footprint: 140k parameters and 193 KB of Int8 Flash, respectively, which is 7–56 $\times$ and 6–40 $\times$ smaller than the other models. Second, ESPNet attains the *highest* foreground recall (0.971), the metric of greatest importance when the objective is to avoid missing diseased tissue. Its IoU (0.489) trails the MobileNetV3-backed models (LR-ASPP 0.595, DeepLabV3+ 0.610); however, those models attain their higher IoU through an ImageNet-pretrained backbone at 7–27 $\times$ the parameters and 6–21 $\times$ the Flash usage, and DeepLabV3+ requires a 16 MB activation arena. Fast-SCNN and the small U-Net are both larger than ESPNet without surpassing it in IoU. Under a memory-first design objective, ESPNet is the Pareto-optimal choice: it is the only model for which its Flash storage is compatible with a low-hundreds-of-kilobyte budget while retaining the highest recall. We adopt it as the Stage 2 segmenter on this basis.

9.3. Classification and End-to-End Results

Table 8 reports the coupled pipeline over all 5002 held-out test crops: per-species classification accuracy and macro-F1 from the Stage 1 classifier and Stage 2 segmentation IoU and recall on the same crops. The end-to-end classification accuracy is 87.8% (macro-F1 0.875), computed with crop weighting over all 5002 test crops; the per-species mean accuracy of the same Stage 1 classifier, used for the baseline comparison in Table 9, is 0.873 (macro-F1 0.871). Classification exceeds 0.90 on peach (0.98), cabbage (0.93), and cherry (0.94), and it is the lowest on celery (0.75, a two-class crop with the fewest test images and

the most visually similar classes). The residual errors fall predominantly between visually similar diseases: On celery, for instance, anthracnose is frequently predicted as early blight (per-class recall 0.58 versus 0.94), which accounts for that species' lower accuracy. The segmentation quality is essentially unchanged when restricted to the *considered* subset (crops for which their disease is also named correctly): The pooled IoU rises marginally from 0.535 to 0.550 and Dice from 0.697 to 0.710, indicating that the two stages succeed and fail largely independently rather than compounding. Figure 4 shows representative end-to-end outputs (input, ground-truth mask, ESPNet mask, and overlay) with the true and predicted labels; the masks visibly enclose the diseased regions while over-covering their boundaries, which is the qualitative signature of the high-recall operating point.

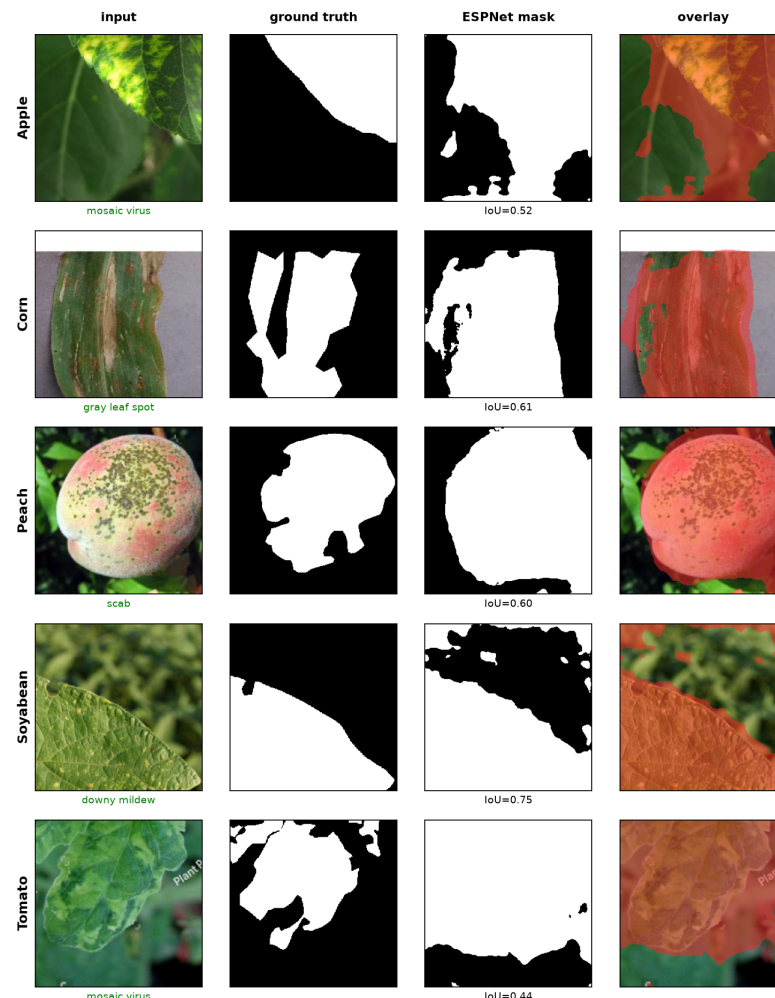


Figure 4. Representative end-to-end outputs on held-out test crops: input, ground-truth mask, ESPNet predicted mask, and overlay. Labels (green) are the true disease with the predicted disease in parentheses; per-crop IoU is shown under each mask. Examples are chosen near each species' mean IoU to reflect typical rather than best-case behaviour.

9.4. Classification Baseline Comparison

Although the classifier derives from prior work, we benchmark it against three widely used ImageNet-pretrained backbones spanning the efficiency spectrum - two similar-scale lightweight classifiers (MobileNetV3-Large and EfficientNet-B0) and one substantially larger model (ConvNeXt-Tiny) - under the identical protocol used for the Stage 1 model (per-species models, balanced training crops with source-grouped validation, held-out test set, 256×256 input, 40 epochs), so all models are trained and evaluated on the same data. Table 9 reports the mean over the 10 species alongside the Int8 footprint. All four

models reach comparable accuracy, yet the per-channel classifier is the most accurate (mean accuracy of 0.873 versus 0.828, 0.825, and 0.837) while being by far the smallest. The operative difference is efficiency: relative to the two similar-scale lightweight backbones, the per-channel classifier uses $10\text{--}13\times$ fewer parameters, $7\text{--}10\times$ smaller Int8 Flash, $7\text{--}12\times$ fewer multiply-accumulates, and $5\text{--}8\times$ less activation memory; it uses roughly $90\times$ fewer parameters than ConvNeXt-Tiny without conceding accuracy; and it is the only one of the four that meets the edge Flash and RAM budget of the target hardware.

Table 8. End-to-end pipeline on the held-out PlantSeg test set (5002 crops, 10 species, and 34 classes). Classification from Stage 1; segmentation from Stage 2 on the same crops.

Plant	Test Crops	Clf. Acc.	Clf. Macro-F1	Seg. IoU	Seg. Recall
Apple	600	0.898	0.897	0.553	0.968
Banana	600	0.842	0.841	0.538	0.983
Bell Pepper	533	0.850	0.839	0.428	0.914
Cabbage	450	0.933	0.933	0.432	0.991
Celery	269	0.751	0.746	0.169	0.969
Cherry	300	0.937	0.936	0.368	0.933
Corn	450	0.860	0.859	0.612	0.986
Peach	450	0.980	0.980	0.595	0.987
Soybean	750	0.876	0.876	0.742	0.994
Tomato	600	0.847	0.846	0.450	0.985
Overall	5002	0.878	0.875	0.535	0.980

Table 9. Classification baselines under a matched protocol: all models are trained on the same balanced crops (source-grouped validation) and evaluated on the same held-out test set (256×256 input, 40 epochs). Footprint is reported after full-integer Int8; RAM is the peak activation arena. ConvNeXt-Tiny did not fully integer-quantise in our export pipeline, so its Int8 Flash and Int8 RAM are omitted (—).

Model	Params	MAdds (M)	Int8 (KB)	RAM (KB)	Acc	Macro-F1
MobileNetV3-Large	3,000,196	289.4	3405.9	2080	0.828	0.827
EfficientNet-B0	4,054,695	523.0	4798.7	3120	0.825	0.823
ConvNeXt-Tiny	27,823,204	5733.3	—	—	0.837	0.833
Chroma-Sense 2.0 (ours)	307,532	43.1	458.8	384	0.873	0.871

9.5. Robustness to Input Blur

Field imagery is subject to defocus and motion blur (for example, from camera shake on a moving platform). To probe robustness to this effect, we re-evaluated the ESPNet segmenter on the held-out test crops after applying Gaussian blur of radius 1 to three pixels (180 crops per species), leaving the ground-truth masks unchanged. Table 10 reports the result. Foreground recall, the operative metric for this recall-oriented design, remains high across all blur levels, declining by at most 0.04 (from 0.97–0.99 to 0.93–0.96) even at radius 3, while IoU degrades only modestly (by ≤ 0.06 ; on corn, it is essentially unchanged). Because the segmenter keys on the broad lesion signature rather than fine boundaries, moderate blur does not materially change which regions are flagged. This gives quantitative support for the robustness implied by the coarse, recall-oriented operating point.

9.6. Effect of Background: Full-Scene vs. Mask-Guided Input

Because the Stage 1 classifier reads the full-scene crop rather than an isolated lesion, a natural question is whether the background helps or hurts and whether restricting the input to the lesion - as a future mask-guided variant might - would improve accuracy. We probed this by re-running the *same* per-species classifiers on the held-out test crops under two mask-derived inputs using the ground-truth lesion mask as an oracle: masked, the crop

with its background zeroed, and crop, a tight bounding box around the mask resized to 256×256 . Table 11 reports the outcome. Both mask-derived inputs *reduce* accuracy relative to the full scene: Overall accuracy falls from 0.878 to 0.807 for the lesion crop and to 0.642 for the background-removed image (macro-F1 $0.875 \rightarrow 0.801 \rightarrow 0.616$). The classifier is thus not distracted by the background; it exploits the whole-crop context, and removing this context at inference degrades it. This is a zero-shot test - the classifier was trained on full scenes, so the mask-derived inputs are out of its training distribution, and the drop reflects that shift as much as any loss of information - and the ground-truth masks make it a best case, since predicted masks could only do worse. The practical implication is that mask-guided cropping is not a free inference-time gain: Realising a benefit would require re-training the classifier on lesion-cropped inputs, which we leave to future work (Section 10). It also confirms that the full-scene setting is a reasonable design choice rather than a handicap.

Table 10. Robustness of the ESPNet segmenter to Gaussian input blur on the held-out test crops (180 crops per species); ground-truth masks are unchanged. Recall stays high, and IoU degrades only modestly up to radius 3 px.

Blur Radius (px)	Apple		Soybean		Corn	
	IoU	Recall	IoU	Recall	IoU	Recall
0 (clean)	0.544	0.971	0.749	0.995	0.605	0.984
1	0.508	0.968	0.734	0.993	0.606	0.981
2	0.488	0.960	0.714	0.981	0.602	0.974
3	0.483	0.944	0.691	0.933	0.599	0.959

Table 11. Effect of restricting the Stage 1 classifier’s input to the lesion on the held-out test set. Full: Full-scene crop; Masked: background zeroed with the ground-truth mask; Crop: tight ground-truth-mask bounding box resized to 256×256 . All three use the same full-scene-trained classifiers, so masked/crop are zero-shot. Values are accuracy.

Plant	Full	Masked	Crop
Apple	0.898	0.630	0.805
Banana	0.842	0.660	0.805
Bell Pepper	0.850	0.445	0.717
Cabbage	0.933	0.511	0.791
Celery	0.751	0.472	0.665
Cherry	0.937	0.810	0.923
Corn	0.860	0.769	0.793
Peach	0.980	0.769	0.891
Soybean	0.876	0.780	0.841
Tomato	0.847	0.537	0.813
Mean	0.877	0.638	0.805
Overall	0.878	0.642	0.807

10. Discussion

10.1. A Recall-Oriented Operating Point

The most visible characteristic of the segmentation stage is the gap between its high foreground recall (0.97) and its moderate IoU (0.49). This is a deliberate consequence of the recall-weighted loss, which penalises missed lesion pixels $20\times$ more than false positives. The resulting masks reliably enclose the diseased tissue but over-cover its boundaries, so precision and hence IoU are lower than a boundary-optimised model would achieve. For a perception stage for which its downstream purpose is to flag *where* disease is present, rather than to trace exact lesion contours, favouring recall over boundary precision is a reasonable

trade-off, since a missed lesion is a more consequential error than an over-covered boundary. We report recall and IoU jointly so that this trade-off is explicit rather than obscured within a single aggregate score.

Crucially, this trade-off is benign for any downstream consumer that acts at a granularity coarser than the boundary error. Figure 5 illustrates this point: A pixel-precise mask and the smoother, slightly over-covering mask produced here span a similar diseased-area fraction, and when the mask is consumed by a coarse actuator, for example, a cell-level treatment system, both masks map to the same coarse outcome, so the boundary shape is immaterial. Trading boundary precision (and hence IoU) for recall therefore does not degrade such downstream use; it merely relocates the residual error to boundary over-coverage, which the coarse consumer absorbs. In this sense, the lower IoU is not a loss of task-relevant information but a controlled and tolerable form of over-segmentation. This actuator argument is conceptual and illustrative: we do not evaluate a physical actuator here, and such validation is left to future work.

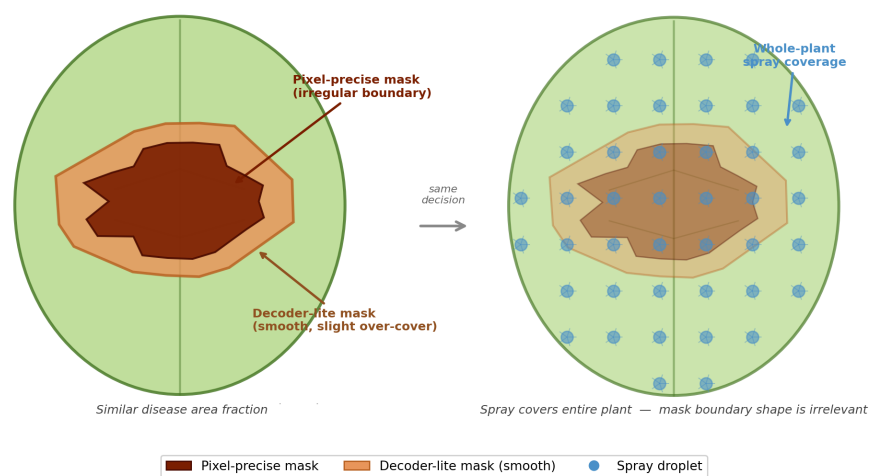


Figure 5. The recall-oriented operating point is benign for a coarse downstream consumer. **(Left):** A pixel-precise mask (dark) and the smoother, slightly over-covering mask produced here (light) span a similar diseased-area fraction. **(Right):** When the mask drives a coarse, cell-level actuator, the boundary shape is immaterial because both masks map to the same set of actuated cells. The recall-for-precision (IoU) trade-off therefore does not degrade such downstream use. This figure is a conceptual illustration of the operating-point argument; validating it on a physical cell-level actuator or spray grid is beyond the scope of this paper and is left to future work.

10.2. Limitations

The pipeline has several acknowledged limitations. The first is related to hardware: At 256×256 , the segmenter's ≈ 4 MB contiguous activation arena exceeds the SRAM of the base H7 and cannot be allocated as a single contiguous block from the roughly 4 MB usable heap of H7 Plus, so on the tested OpenMV firmware, the segmenter does not run on either board. The classifier runs on both, but the full pipeline currently needs a runtime that maps the arena directly into SDRAM or a single-board computer. The second limitation is related to the task: The classifier (Stage 1) operates on the full-scene crop rather than an isolated lesion, so scenes containing more than one disease may yield an averaged or ambiguous label; the per-channel extractor assumes a single dominant disease per crop. Restricting the input to the lesion does not itself help: The full-scene classifier relies on the whole-crop context, and masking or cropping to the lesion lowers accuracy (Section 9.6). The third is related to the IoU: The coarse, recall-oriented masks are not suited to applications that require exact lesion shape. The fourth limitation is related to scale: After cropping and balancing, each disease class has on the order of a few hundred training crops drawn

from a smaller pool of underlying source images, so overfitting is a real risk. We limit this by pre-training a shared 34-class representation before per-species fine-tuning, source-grouped train/validation splitting, inverse-frequency class weighting, augmentation-based balancing, and early stopping on validation loss; even so, this data scale is sufficient to rank architectures and demonstrate the pipeline but not to claim production-level generalisation, and larger field-collected sets (Section 10, future work) are needed to establish it.

10.3. Future Work

Three extensions follow directly. (i) The segmentation mask could seed *mask-guided cropping*, running the two stages in the reverse (segment-then-classify) order so that the classifier re-examines only the flagged regions, which may sharpen classification on multi-disease scenes at little extra cost; our zero-shot analysis (Section 9.6) indicates that this would require re-training the classifier on lesion-cropped inputs to be effective, since the full-scene classifier alone loses accuracy on cropped inputs. (ii) A smaller input resolution or *structured* channel would shrink the segmenter's activation arena enough to target smaller microcontrollers, and an on-device energy measurement would quantify the power cost of each stage. (iii) Larger, field-collected training sets would test how far the reported accuracy generalises. Finally, the coarse-actuation benefit illustrated in Figure 5 is argued conceptually; validating it on a physical cell-level actuator or spray grid is a further direction.

11. Conclusions

We presented Chroma-Sense 2.0, a two-stage on-device pipeline for plant–disease perception that decouples binary lesion segmentation from disease classification. Stage 1 is an updated per-channel classifier, and Stage 2 is a compact ESPNet segmenter; the two run sequentially on the same frame, so the pipeline's peak activation memory is bounded by the larger stage rather than their sum. On a leakage-controlled, 10-species, 34-class subset of the in-the-wild PlantSeg dataset, ESPNet attains a mean foreground recall of 0.97 (mean IoU 0.49 m, pooled 0.53) and is the smallest-footprint model (140k parameters, 193 KB Int8 Flash) among lightweight segmentation baselines while retaining the highest recall. End to end, the coupled pipeline names the disease correctly on 87.8% of 5002 held-out test crops. The on-device profile shows that, at 256×256 , the binding constraint is the segmenter's ≈ 4 MB contiguous activation arena, not the 651.9 KB of combined Flash. On the 32 MB-SDRAM H7 Plus, the usable MicroPython heap is only about 4 MB, and this arena cannot be allocated as a single contiguous block from it, so on the tested firmware, the classifier runs on-device while the segmenter needs either a runtime that maps the arena directly into SDRAM or a single-board computer. In practice, the classification stage already runs on microcontroller-class boards, while the full pipeline fits the gigabyte-scale single-board-computer tier, such as a Raspberry Pi or NVIDIA Jetson Nano. Reducing the segmenter's peak activation so that full-resolution segmentation also reaches the microcontroller tier is therefore the primary deployment direction.

Author Contributions: Conceptualization, K.K.K.; methodology, K.K.K.; software, K.K.K. and A.T.; validation, K.K.K. and A.T.; writing—original draft, K.K.K.; writing—review and editing, S.P.M., A.T., and E.K.; supervision, S.P.M. and E.K. All authors have read and agreed to the published version of this manuscript.

Funding: This research received no external funding.

Data Availability Statement: The PlantSeg dataset used in this study is publicly available. Code and trained model weights are available from the corresponding author upon reasonable request.

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

References

- Mitra, A.; Vangipuram, S.L.T.; Bapatla, A.K.; Bathalapalli, V.K.V.V.; Mohanty, S.P.; Kougianos, E.; Ray, C. Everything You Wanted to Know About Smart Agriculture. *arXiv* **2022**, arXiv:2201.04754.
- Zhang, X.; Cao, Z.; Dong, W. Overview of Edge Computing in the Agricultural Internet of Things: Key Technologies, Applications, Challenges. *IEEE Access* **2020**, *8*, 141748–141761. [\[CrossRef\]](#)
- Varghese, B.; Wang, N.; Barbhuiya, S.; Kilpatrick, P.; Nikolopoulos, D.S. Challenges and Opportunities in Edge Computing. In Proceedings of the 2016 IEEE International Conference on Smart Cloud (SmartCloud), New York, NY, USA, 18–20 November 2016; pp. 20–26. [\[CrossRef\]](#)
- 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.
- Howard, A.; Sandler, M.; Chu, G.; Chen, L.C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for MobileNetV3. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 1314–1324. [\[CrossRef\]](#)
- Tan, M.; Le, Q.V. EfficientNetV2: Smaller Models and Faster Training. *arXiv* **2021**, arXiv:2104.00298.
- Zhang, X.; Zhou, X.; Lin, M.; Sun, J. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 6848–6856. [\[CrossRef\]](#)
- Li, Y.; Nie, J.; Chao, X. Do We Really Need Deep CNN for Plant Diseases Identification? *Comput. Electron. Agric.* **2020**, *178*, 105803. [\[CrossRef\]](#)
- Wei, K.; Chen, B.; Zhang, J.; Fan, S.; Wu, K.; Liu, G.; Chen, D. Explainable Deep Learning Study for Leaf Disease Classification. *Agronomy* **2022**, *12*, 1035. [\[CrossRef\]](#)
- Kethineni, K.K.; Wu, S.Y.; Mohanty, S.P.; Kougianos, E. Chroma-Sense: A Memory-Efficient Plant Leaf Disease Classification Model for Edge Devices. In Proceedings of the 2025 IEEE Conference on Artificial Intelligence (CAI), Santa Clara, CA, USA, 5–7 May 2025; pp. 1–6. [\[CrossRef\]](#)
- Mehta, S.; Rastegari, M.; Caspi, A.; Shapiro, L.; Hajishirzi, H. ESPNet: Efficient Spatial Pyramid of Dilated Convolutions for Semantic Segmentation. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018. [\[CrossRef\]](#)
- Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Proceedings of the Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015*; Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F., Eds.; Springer International Publishing: Cham, Switzerland, 2015; Volume 9351, pp. 234–241. [\[CrossRef\]](#)
- Chen, L.C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In *Proceedings of the Computer Vision—ECCV 2018*; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Springer International Publishing: Cham, Switzerland, 2018; Volume 11211, pp. 833–851. [\[CrossRef\]](#)
- Yu, C.; Wang, J.; Peng, C.; Gao, C.; Yu, G.; Sang, N. BiSeNet: Bilateral Segmentation Network for Real-Time Semantic Segmentation. In *Proceedings of the Computer Vision—ECCV 2018*; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Springer International Publishing: Cham, Switzerland, 2018; Volume 11217, pp. 334–349. [\[CrossRef\]](#)
- 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 (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520. [\[CrossRef\]](#)
- Tan, M.; Le, Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In Proceedings of the 36th International Conference on Machine Learning (ICML), Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
- Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-Level Accuracy with 50x Fewer Parameters and Less than 0.5 MB Model Size. *arXiv* **2016**, arXiv:1602.07360.
- Lin, J.; Chen, W.M.; Lin, Y.; Cohn, J.; Gan, C.; Han, S. MCUNet: Tiny Deep Learning on IoT Devices. In *Proceedings of the Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*; Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2020; Volume 33, pp. 11452–11464.
- Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Li, F.-F. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Miami, FL, USA, 20–25 June 2009; pp. 248–255. [\[CrossRef\]](#)
- Mohanty, S.P.; Hughes, D.P.; Salathé, M. Using Deep Learning for Image-Based Plant Disease Detection. *Front. Plant Sci.* **2016**, *7*, 1419. [\[CrossRef\]](#) [\[PubMed\]](#)
- Archana, U.; Khan, A.; Sudarshanam, A.; Sathya, C.; Koshariya, A.K.; Krishnamoorthy, R. Plant Disease Detection Using ResNet. In Proceedings of the 2023 International Conference on Inventive Computation Technologies (ICICT), Lalitpur, Nepal, 26–28 April 2023; pp. 614–618. [\[CrossRef\]](#)

22. Ferentinos, K.P. Deep Learning Models for Plant Disease Detection and Diagnosis. *Comput. Electron. Agric.* **2018**, *145*, 311–318. [[CrossRef](#)]
23. Ramcharan, A.; Baranowski, K.; McCloskey, P.; Ahmed, B.; Legg, J.; Hughes, D.P. Deep Learning for Image-Based Cassava Disease Detection. *Front. Plant Sci.* **2017**, *8*, 1852. [[CrossRef](#)] [[PubMed](#)]
24. Peker, M. Multi-Channel Capsule Network Ensemble for Plant Disease Detection. *SN Appl. Sci.* **2021**, *3*, 707. [[CrossRef](#)]
25. Rakib, A.F.; Rahman, R.; Razi, A.A.; Hasan, A.S.M.T. A Lightweight Quantized CNN Model for Plant Disease Recognition. *Arab. J. Sci. Eng.* **2024**, *49*, 4097–4108. [[CrossRef](#)]
26. Bedi, P.; Gole, P. Plant Disease Detection Using Hybrid Model Based on Convolutional Autoencoder and Convolutional Neural Network. *Artif. Intell. Agric.* **2021**, *5*, 90–101. [[CrossRef](#)]
27. Chowdhury, M.E.H.; Rahman, T.; Khandakar, A.; Ayari, M.A.; Khan, A.U.; Khan, M.S.; Al-Emadi, N.; Reaz, M.B.I.; Islam, M.T.; Ali, S.H.M. Automatic and Reliable Leaf Disease Detection Using Deep Learning Techniques. *AgriEngineering* **2021**, *3*, 294–312. [[CrossRef](#)]
28. Ashwinkumar, S.; Rajagopal, S.; Manimaran, V.; Jegajothi, B. Automated Plant Leaf Disease Detection and Classification Using Optimal MobileNet Based Convolutional Neural Networks. *Mater. Today Proc.* **2022**, *51*, 480–487. [[CrossRef](#)]
29. Barbedo, J.G.A. Plant Disease Identification from Individual Lesions and Spots Using Deep Learning. *Biosyst. Eng.* **2019**, *180*, 96–107. [[CrossRef](#)]
30. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 936–944. [[CrossRef](#)]
31. Poudel, R.P.K.; Liwicki, S.; Cipolla, R. Fast-SCNN: Fast Semantic Segmentation Network. In Proceedings of the British Machine Vision Conference (BMVC), Cardiff, UK, 9–12 September 2019.
32. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Proceedings of the Advances in Neural Information Processing Systems 28 (NIPS 2015)*; Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2015; Volume 28, pp. 91–99.
33. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788. [[CrossRef](#)]
34. Wei, T.; Chen, Z.; Yu, X.; Chapman, S.; Melloy, P.; Huang, Z. PlantSeg: A Large-Scale In-the-Wild Dataset for Plant Disease Segmentation. *Sci. Data* **2026**, *13*, 205. [[CrossRef](#)] [[PubMed](#)]
35. Wei, T. *PlantSeg: A Large-Scale In-the-Wild Dataset for Plant Disease Segmentation*; Zenodo: Geneva, Switzerland, 2024. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.