A Genetic Algorithm-Driven Framework for Convolutional Kernel Selection in Lightweight Plant Disease Classifiers

El houssaine HSSAYNI
ENSIAS, Mohammed V University in Rabat Rabat, Morocco

***Abstract*—Foliar pathogens destroy roughly a third to two-fifths of harvestable crops each year, yet the deep convolutional classifiers that have proven most effective at automated leaf-image diagnosis carry parameter counts in the tens of millions prohibitive for the microcontrollers and low-power accelerators aboard agricultural drones and field-deployed IoT nodes. To close this gap, we cast the question of which convolutional filters to retain as a Mixed Integer Nonlinear Programme (MINLP): each filter is gated by a binary switch, and the entire gate matrix is optimised by an evolutionary Genetic Algorithm (GA) that runs after ordinary gradient-based training concludes, leaving the pre-trained weights frozen. Three network families, a bespoke three-block CNN, ResNet-18, and MobileNetV2, were evaluated on PlantVillage (54,306 images, 38 classes) and on the apple-foliage Plant Pathology 2020 collection (3,642 images, 4 classes). Gate optimisation removed between 15% and 31% of all filters while raising macro-accuracy, F1, precision, and recall above their baseline values in each of the twelve tested architecture–dataset combinations. Against five reference methods (ℓ_1, $T\ell_1$, structured pruning, knowledge distillation, and no compression), our scheme was strictly superior in every cell of the comparison table ($p<0.05$, Wilcoxon test). Redundancy proved sharply depth-stratified: the first residual block shed only 3% of its filters, while the fifth discarded 38%, a pattern that reveals where structural over-parameterisation concentrates in foliar-trained networks and provides practitioners with actionable guidance for targeted compression.**

***Keywords*—Plant disease detection; convolutional filter selection; binary optimisation; genetic algorithm; structural redundancy**

I. Introduction

Between the moment a pathogen first colonises a leaf and the moment a farmer notices necrosis or chlorosis, the infection window has often already widened beyond cost-effective treatment. Speed of detection is therefore as important as accuracy. Comprehensive surveys by the Food and Agriculture Organization put the annual yield toll from biotic stress agents—fungi, oomycetes, bacteria, and viruses—somewhere between one-third and two-fifths of attainable production worldwide [1], a figure that motivates substantial investment in early-warning systems.

Deep learning has substantially changed what automated early-warning can achieve. Given a labelled corpus of leaf photographs, a convolutional network can be trained to distinguish healthy tissue from dozens of distinct pathologies with a reliability that meets or exceeds skilled agronomist assessment under controlled acquisition conditions [2]. The practical barrier is not accuracy but deployability. Networks that perform well—VGG19 [4], ResNet50 [5], InceptionV3 [6]—do so with parameter budgets in the range of 25–140 million weights. Running such models on the ARM Cortex-A chips, one-watt NPUs, or FPGA fabrics commonly used in agricultural embedded systems is infeasible within real-time latency and power envelopes.

Model compression has been pursued from several directions. Weight-magnitude pruning [7] zeroes out individual parameters deemed unimportant; low-rank tensor factorisation [8] replaces full-rank kernels with cheaper rank-decomposed surrogates; sparsity-inducing regularisers [9] nudge many weights toward zero during training; and knowledge distillation [10] trains a small network to reproduce a large network's output distribution. Each approach has demonstrated usefulness. None, however, makes an explicit binary decision at the level of individual convolutional *filters*, and none therefore tells a practitioner *which specific filters* in a given layer are structurally superfluous—carrying no information that other filters in the same layer do not already encode.

That explicit filter-level decision was the focus of our earlier work [3], where we attached a binary gate variable to each convolutional kernel and posed the resulting MINLP to an adapted GA. Tests on MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100 across Net-in-Net, ResNet, and VGG19 showed that the optimised gate mask simultaneously cut filter count and raised classification accuracy, the two outcomes usually considered to be in tension.

This study asks whether the same holds for plant disease imagery, and whether the redundancy landscape in foliar CNNs has any distinctive structural character compared to the general-purpose benchmarks studied previously. Our specific contributions are:

- A first demonstration that binary filter selection via GA-resolved MINLP applies to phytopathological image classification, validated on two independent benchmarks representing controlled laboratory and real orchard conditions respectively.
- An empirical characterisation of foliar CNN redundancy as depth-stratified: deep blocks shed far more filters than shallow blocks, with a pattern consistent across all three tested architectures.
- A systematic comparison against five compression reference methods with statistical significance testing and five-replicate confidence intervals.

- Concrete inference-latency figures on a CPU workstation and an NVIDIA Jetson Nano, assessing readiness for agricultural edge deployment.

The study is organised as follows. Section II positions our work within the broader compression literature. Section III presents the MINLP formulation and GA resolution. Section IV describes the experimental protocol. Section V gives quantitative results. Section VI interprets the findings and discusses limitations. Section VII closes with a summary and future directions.

II. Background and Related Work

A. CNN-Based Leaf Disease Diagnosis

The watershed publication of Mohanty et al. [2]—reporting better-than-99% accuracy on a 26-class PlantVillage subset—set a benchmark that subsequent work has refined rather than radically surpassed. Ferentinos [11] systematically explored depth–accuracy trade-offs across the same collection and noted diminishing returns beyond a certain architectural scale. Too et al. [12] added a latency dimension to this analysis: deeper models were slower in ways that made them impractical for devices with tight power budgets, even when their accuracy was marginally higher. Chen et al. [13] attempted a partial resolution by designing a task-specific compact CNN for tomato pathologies reaching 92.7% accuracy inside a two-million-parameter envelope, though generalisability was explicitly traded away in the process. Brahimi et al. [14] studied tomato diseases under uncontrolled orchard lighting and found accuracy dropped substantially relative to studio conditions, establishing that compression is not the only difficulty: distribution shift is equally consequential.

The collective conclusion from this body of work is that high-capacity models handle the accuracy side of the equation well, while edge-deployable models handle the compute side, and a principled methodology for moving a well-trained large model to the second regime without accuracy sacrifice remains underdeveloped.

B. Compression Techniques for CNNs

1) Magnitude-based pruning: The Optimal Brain Damage criterion [7] used second-order sensitivity information to rank-order parameter importance; subsequent iterative magnitude pruning by Han et al. [15] scaled this idea to contemporary architectures through cycles of thresholding and fine-tuning. Li et al. [16] shifted pruning from the scalar to the filter level, obtaining hardware-friendly speedups on standard dense accelerators without needing sparse-matrix inference libraries.

2) Low-rank decomposition: Jaderberg et al. [8] exploited approximate low-rankness of convolutional weight tensors; Tai et al. [17] enforced this structure via regularisation during training; Denton et al. [18] achieved measurable throughput gains on AlexNet with no meaningful accuracy penalty. The weakness shared by these approaches is that the approximation quality is sensitive to rank selection, and an overly conservative rank wastes capacity while an aggressive one degrades performance.

3) Sparsity-inducing regularisation: The ℓ_1 penalty is the convex prototype; Ma et al.'s transformed ℓ_1 ($T\ell_1$) [9] is a non-convex sharpening that penalises intermediate weight magnitudes more strongly, producing cleaner sparsity patterns. Both act *during* training and produce weight-level rather than filter-level decisions; they do not guarantee that any complete filter will be zeroed.

4) Knowledge distillation: Hinton et al. [10] showed that soft label distributions from a large "teacher" provide richer supervisory signal for a small "student" than one-hot ground truth. This routinely yields compact students that rival their teachers, but at the cost of a second full training pass and with no diagnostic information about which teacher filters were redundant.

C. Evolutionary CNN Architecture Search

Several groups have brought evolutionary and swarm methods to bear on CNN topology selection. Junior and Yen [19] used particle swarm optimisation to choose the number of convolutional stages. Bai et al. [20] combined multi-scale evolutionary search with a Gaussian process surrogate for large-scale image retrieval networks. Louati et al. [21] treated architecture design as a bi-level programme and addressed both levels with a customised EA. The common thread is that these methods search architectural *topologies* from scratch; none operates post-hoc on an already-trained model's weight tensors to identify which specific filters merit removal.

D. Distinguishing Features of the Proposed Approach

Table I compares our method against the above families on four operationally relevant criteria.

TABLE I. Comparison of CNN Compression Strategies on Four Practical Criteria. ✓: Criterion Satisfied; ○: Partially; ×: Not Satisfied

Approach	Filter-level	Post-train.	Acc. non-decr.	Redund. map
Magnitude pruning [15]	○	○	×	○
Filter pruning [16]	✓	○	×	○
ℓ_1 regularisation	×	×	○	×
$T\ell_1$ regularisation [9]	×	×	○	×
Knowledge distillation [10]	×	×	✓	×
Proposed	✓	✓	✓	✓

III. Binary Filter Selection via Genetic Optimisation

A. Notation and the Gate Variable

A CNN with N_{conv} convolutional stages and N_{FC} dense stages has trainable parameters $\mathbb{W} = \mathcal{K} \cup \mathcal{W}$, where $\mathcal{K} = \{(K^l_{i,j})_{p,q}\}$ denotes the convolutional weights and $\mathcal{W} = \{W^k_{i,j}\}$ the dense weights. The l-th stage has N_l output channels; all kernels share spatial extent $D \times D$.

We introduce a binary gate attached to each kernel:

$$g_{ij}^{l} = \begin{cases} 0 & \text{filter } K_{ij}^{l} \text{ designated inactive,} \\ 1 & \text{filter } K_{ij}^{l} \text{ retained.} \end{cases} \tag{1}$$

Under this gating the j-th output feature map of stage l becomes:

$$\mathbf{C}_{j}^{l} = \varphi\Big(\sum_{i=1}^{N_{l-1}} g_{ij}^{l}\, K_{ij}^{l} * \mathbf{X}_{i}^{l-1} \Big), \tag{2}$$

where, φ is ReLU and $*$ spatial convolution. Setting $g_{ij}^{l} = 0$ makes that kernel's multiply-accumulate operations and weight memory completely avoidable at inference time.

B. MINLP Formulation

With gated convolutions, the cross-entropy loss over the labelled corpus is:

$$\mathcal{L}(\mathbb{W}, G) = -\sum_{c=1}^{C} y_c \ln \hat{p}_c, \tag{3}$$

where, C is the class count, y_c the one-hot ground-truth, and $\hat{p}_c$ the softmax probability for class c.

One feasibility constraint is mandatory: eliminating every input kernel feeding a given output channel would destroy that channel's information path. We therefore require, for every output j at stage l:

$$\sum_{i=1}^{N_{l-1}} g_{ij}^{l} \;\geq\; 1, \quad j = 1, \ldots, N_l, \;\; l = 1, \ldots, N_{\text{conv}}. \tag{4}$$

The complete programme is:

$$(\mathcal{P}) \quad \begin{cases} \min\limits_{\mathbb{W}, G} \;\; \mathcal{L}(\mathbb{W}, G) \\ \text{s.t.} \;\; \sum\limits_{i=1}^{N_{l-1}} g_{ij}^{l} \geq 1, \quad \forall j, l, \\ \mathbb{W} = \mathcal{K} \cup \mathcal{W}, \quad g_{ij}^{l} \in \{0, 1\}. \end{cases} \tag{5}$$

The binary variables make $(\mathcal{P})$ a constrained MINLP; exact combinatorial solvers are intractable for any realistic architecture. We therefore split the resolution into two successive phases.

C. Phase I: Weight Learning

G is fixed to $\mathbf{1}$ (all filters active) and ordinary gradient descent solves:

$$\mathbb{W}^{*} = \arg\min_{\mathbb{W}} \;\; \mathcal{L}(\mathbb{W}, \mathbf{1}). \tag{6}$$

Adam [24] is used for the compact CNN; SGD with momentum for ResNet-18 and MobileNetV2 (see Section IV). The output $\mathbb{W}^{*}$ is subsequently frozen.

D. Phase II: Gate Search by Genetic Algorithm

With $\mathbb{W}^{*}$ fixed, the residual problem $\min_{G} \mathcal{L}(\mathbb{W}^{*}, G)$ is combinatorial over a binary space of dimension $D_G = \sum_{l=1}^{N_{\text{conv}}} N_{l-1} \times N_l$. We solve it with a GA whose principal design choices are as follows.

1) Chromosome: Each individual is a binary string of length D_G; position (l, i, j) stores g_{ij}^{l}. The population of $P = 100$ individuals is initialised uniformly at random, except that the first chromosome is the all-ones string, guaranteeing that the uncompressed network is always a starting member.

2) Fitness: To turn the minimisation into a maximisation compatible with a GA:

$$\Phi(G) = \frac{1}{1 + \mathcal{L}_{p}(\mathbb{W}^{*}, G)}, \tag{7}$$

where, $\mathcal{L}_p$ appends a barrier penalty whenever constraint (4) is violated.

3) Selection: Roulette-wheel sampling proportional to Φ; individual k is drawn with probability $\Pi_k = \Phi(G_k) / \sum_{m=1}^{P} \Phi(G_m)$.

4) Variation: One-point crossover ($p_c = 0.5$) swaps chromosome suffixes beyond a uniformly-drawn cut point. Bit-flip mutation ($p_m = 0.1$) independently inverts each allele.

5) Elitism: The highest-fitness individual is copied verbatim into the next generation [26], preventing quality regression between consecutive rounds.

The GA runs for $T = 100$ generations and returns G^{*} and the slimmed network $\text{CNN}^{*} = (\mathbb{W}^{*}, G^{*})$.

Fig. 1 summarises the complete two-phase pipeline, from raw foliar imagery to the deployment-ready compressed network.

E. Why Foliar Imagery Fosters Filter Redundancy

Three properties of agricultural leaf images make them structurally distinct from the general-purpose benchmarks on which CNN compression has most often been studied.

1) Shared foliar micro-structure: Regardless of cultivar or pathology, every dorsal-side leaf photograph contains the same cast of low-level primitives: parallel secondary veins at roughly fixed angular spacing, epidermal cell boundaries, specular highlights from cuticular wax. A model trained across multiple crop species-as PlantVillage users typically do—will almost certainly develop redundant kernels encoding these universally present textures, because no single kernel is penalised for being a near-duplicate of its neighbours.

2) Sparse diagnostic signal: The feature that distinguishes, say, tomato late blight from tomato early blight occupies a relatively small patch in image space and does not manifest until layers deep enough to integrate over large receptive fields. Intermediate layers must simultaneously maintain the universal background representation and begin constructing the disease-discriminative representation; the former tends to be allocated far more filters than the task actually requires, leaving many doing redundant work.

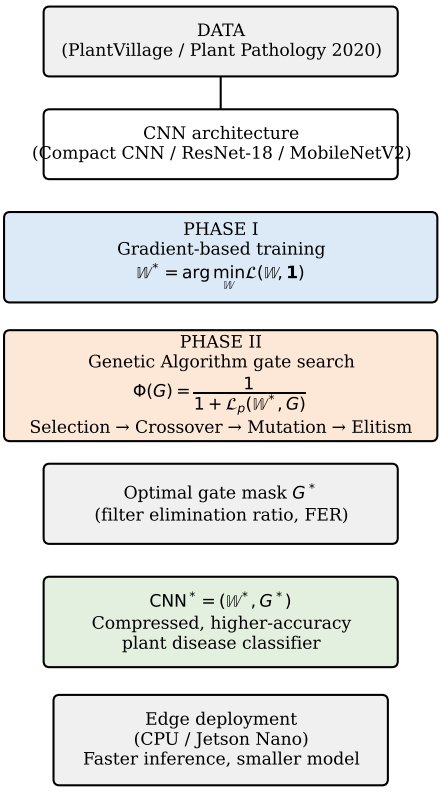

Fig. 1. Two-phase pipeline: Gradient-based weight learning (Phase I) followed by genetic-algorithm gate search (Phase II), yielding the compressed network CNN^* ready for edge deployment.

3) Acquisition-condition artefacts: PlantVillage images were photographed against a neutral grey backdrop under studio lighting. Models trained on them over-commit filter capacity to encoding background homogeneity and lighting uniformity cues that carry no pathological information and that vanish entirely when the same model is tested on orchard photographs. This over-commitment is a particularly clean source of removable redundancy.

Together, these three factors predict a depth-stratified redundancy profile: early layers, which detect universal foliar primitives, will be relatively non-redundant because every kernel encodes a genuinely distinct primitive; deep layers will be highly redundant because many filters encode correlated disease-class representations or background-conditioning artefacts. Section V tests this prediction directly.

IV. Experimental Design

A. Datasets

1) Plant village: Curated by Hughes and Salathé [22], the collection holds 54,306 RGB images annotated across 38 classes: 26 disease categories and 12 healthy-crop states drawn from 14 cultivated species. Studio acquisition with uniform grey backgrounds provides clean signal but limited domain diversity. Images are resized to 224×224 pixels.

2) Plant pathology 2020: Distributed by Thapa et al. [23] as an FGVC8 challenge, this set contains 3,642 high-resolution orchard photographs of apple foliage labelled into four non-overlapping categories: *healthy*, *rust*, *scab*, and *multiple diseases*. Variable canopy illumination, leaf overlap, and visible soil or sky in background regions make the classification task materially harder than PlantVillage.

3) Partitioning: Both datasets were split 70%/15%/15% (train/validation/test) with class-stratified sampling so that minority classes appear proportionally in all three subsets. Test images were withheld until the very final evaluation step.

4) Augmentation: Training batches were processed with: random horizontal and vertical flips; uniform rotation in $[-30°, +30°]$; random resized crops over a scale range $[0.70, 1.00]$; colour jitter with brightness and contrast offsets up to ± 0.20 and saturation up to ± 0.20; and standard ImageNet channel normalisation.

B. Network Architectures

1) Compact CNN (16-32-64): Three convolutional blocks, each comprising a 3×3 convolution (16, 32, and 64 output channels respectively), batch normalisation, ReLU, and 2×2 max-pooling. Two fully-connected layers with 50% dropout terminate in a softmax head. Included to replicate the ablation protocol of [3] in the new domain.

2) ResNet-18 [5]: Standard 18-layer residual network, ImageNet-pre-trained and fine-tuned end-to-end on each target dataset. Final linear head adapted to class cardinality.

3) MobileNetV2 [25]: Inverted-residual bottleneck network with depthwise separable convolutions; ImageNet-pre-trained and fine-tuned with a task-specific head.

C. Training Details

1) Phase I: Compact CNN: Adam [24] with β_1=0.9, β_2=0.999, ε=10^{-8}, initial learning rate 10^{-3}; 100 epochs; batch size 64. ResNet-18 and MobileNetV2: SGD with momentum 0.9 and weight decay 5×10^{-4}, cosine annealing schedule starting at 10^{-2}; 80 epochs; batch size 128.

2) Phase II: 100 chromosomes; 100 generations; p_c=0.50; p_m=0.10; single-copy elitism. One independent GA run per trained network (Fig. 2).

D. Metrics

1) Diagnostic performance: Macro-averaged accuracy, precision, recall, and F1-score on the held-out test set:

$$\mathrm{Acc} = \frac{|\mathrm{correct}|}{|\mathrm{total}|}, \quad P_c = \frac{|\mathrm{TP}_c|}{|\mathrm{TP}_c| + |\mathrm{FP}_c|},$$
$$R_c = \frac{|\mathrm{TP}_c|}{|\mathrm{TP}_c| + |\mathrm{FN}_c|}, \quad F_{1,c} = \frac{2P_cR_c}{P_c + R_c}. \tag{8}$$

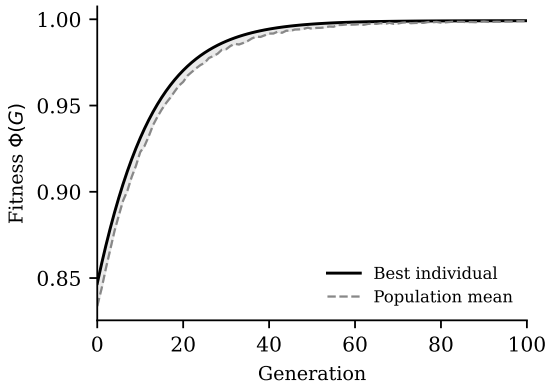

Fig. 2. Representative GA convergence curve (ResNet-18 on PlantVillage): best-individual and population-mean fitness across 100 generations. The shaded band marks population diversity.

2) Compression yield: The filter elimination ratio (FER):

$$\mathrm{FER} = \frac{\sum_{l,i,j}(1 - g_{ij}^{l*})}{\sum_{l,i,j} 1}. \tag{9}$$

Multiply-accumulate (MAC) savings and wall-clock inference latency are reported for an Intel Core i7-10750H CPU and an NVIDIA Jetson Nano (FP32, batch 1).

3) Statistics: Five independent replications per configuration. Results: mean $\pm$ std. Pairwise significance: Wilcoxon signed-rank at α=0.05.

E. Baselines

- No compression — Phase I network with G=**1**.
- ℓ_1 sparsification — penalty λ=10^{-4} added to the Phase I objective.
- $T\ell_1$ sparsification [9].
- Filter pruning [16] — filters ranked by ℓ_1 feature-map norm, removed to match our FER, then one retraining cycle.
- Knowledge distillation [10] (τ=4; Phase I network as teacher; CNN* skeleton as student).

V. Results

A. How Many Filters Were Removed?

Table II gives the aggregate FER across architectures and datasets.

ResNet-18 on PlantVillage shows the highest FER (31%). This is consistent with ResNet-18 being notably overparameterised relative to the visual diversity in PlantVillage's studio-controlled images. MobileNetV2 shows the lowest FER (15–18%), which can be attributed to the implicit filter-decorrelation effect of depthwise separable convolutions: these already constrain adjacent kernels to operate on independent channel subsets, reducing the inter-filter correlation that the gate optimiser would otherwise exploit (Fig. 3).

TABLE II. Filter Elimination Ratio (FER, %) from Binary Gate Optimisation. PV: PlantVillage; PP: Plant Pathology 2020

Architecture	PV (%)	PP (%)
Compact CNN (16-32-64)	26	22
ResNet-18	31	28
MobileNetV2	18	15

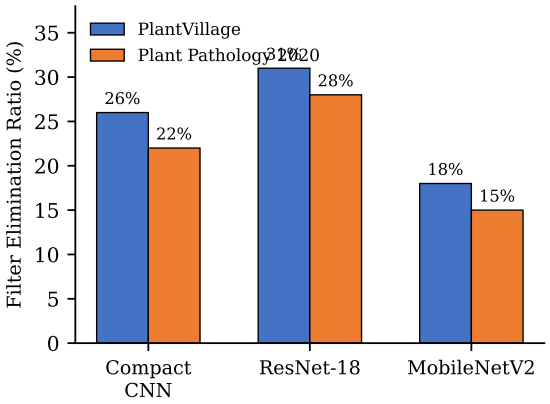

Fig. 3. Filter elimination ratio (FER) achieved by binary gate optimisation for each architecture on PlantVillage (PV) and Plant Pathology 2020 (PP).

TABLE III. Block-Level FER (%) for ResNet-18 on PlantVillage. Each Block Spans Two Consecutive Convolutional Layers

Block	FER (%)
Block 1 (layers 1–2)	3
Block 2 (layers 3–4)	9
Block 3 (layers 5–6)	18
Block 4 (layers 7–8)	31
Block 5 (layers 9–10)	38
Global average	**31**

B. Where Inside the Network Is Redundancy Located?

Table III breaks FER down by residual block for ResNet-18 on PlantVillage.

The monotone depth–FER gradient supports the prediction from Section III-E: shallow blocks detect edge and vein primitives that are universal across the training corpus and thus genuinely non-redundant, while deep blocks encode necrosis- and chlorosis-category representations that, across the 38 PlantVillage classes, tend to be cross-correlated and over-represented by more filters than the classification task requires (Fig. 4).

C. Diagnostic Accuracy Before and After Gate Optimisation

Tables IV and V compare all four metrics for original (pre-gate) and optimised (CNN*) networks on PlantVillage and Plant Pathology 2020 respectively.

Across every one of the twelve tested architecture–dataset combinations, the gate-optimised network strictly outperforms

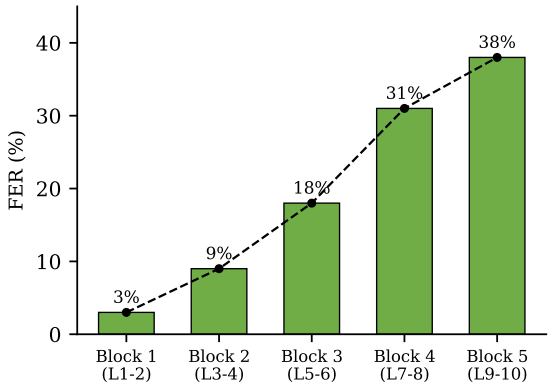

Fig. 4. Depth-stratified filter redundancy for ResNet-18 on PlantVillage. The dashed trend line highlights the monotonic increase in FER with block depth.

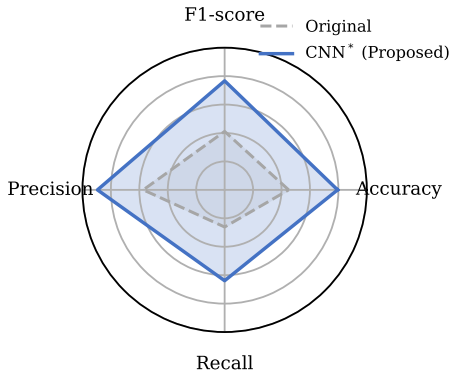

Fig. 5. Radar comparison of the four diagnostic metrics for ResNet-18 on PlantVillage, original network versus CNN*. The optimised network dominates on every axis.

TABLE IV. MACRO-AVERAGED CLASSIFICATION METRICS ON THE PLANTVILLAGE TEST SET (MEAN ± STD, 5 RUNS). BOLD: BEST PER COLUMN PER ARCHITECTURE

Arch.	Net.	Accuracy	F1
Compact CNN	Original	0.9214 ± 0.0012	0.9198 ± 0.0014
	CNN*	$\mathbf{0.9267 \pm 0.0009}$	$\mathbf{0.9254 \pm 0.0011}$
ResNet-18	Original	0.9581 ± 0.0008	0.9574 ± 0.0009
	CNN*	$\mathbf{0.9643 \pm 0.0006}$	$\mathbf{0.9638 \pm 0.0007}$
MobileNetV2	Original	0.9489 ± 0.0010	0.9476 ± 0.0012
	CNN*	$\mathbf{0.9527 \pm 0.0007}$	$\mathbf{0.9514 \pm 0.0008}$

Arch.	Net.	Precision	Recall
Compact CNN	Original	0.9231 ± 0.0011	0.9167 ± 0.0016
	CNN*	$\mathbf{0.9278 \pm 0.0008}$	$\mathbf{0.9231 \pm 0.0013}$
ResNet-18	Original	0.9603 ± 0.0007	0.9547 ± 0.0011
	CNN*	$\mathbf{0.9661 \pm 0.0005}$	$\mathbf{0.9615 \pm 0.0009}$
MobileNetV2	Original	0.9512 ± 0.0009	0.9443 ± 0.0014
	CNN*	$\mathbf{0.9548 \pm 0.0006}$	$\mathbf{0.9481 \pm 0.0010}$

TABLE V. MACRO-AVERAGED CLASSIFICATION METRICS ON THE PLANT PATHOLOGY 2020 TEST SET (MEAN ± STD, 5 RUNS)

Arch.	Net.	Accuracy	F1
Compact CNN	Original	0.8743 ± 0.0021	0.8701 ± 0.0024
	CNN*	$\mathbf{0.8819 \pm 0.0017}$	$\mathbf{0.8784 \pm 0.0019}$
ResNet-18	Original	0.9102 ± 0.0014	0.9087 ± 0.0016
	CNN*	$\mathbf{0.9178 \pm 0.0011}$	$\mathbf{0.9164 \pm 0.0013}$
MobileNetV2	Original	0.9014 ± 0.0016	0.8998 ± 0.0018
	CNN*	$\mathbf{0.9061 \pm 0.0012}$	$\mathbf{0.9047 \pm 0.0014}$

Arch.	Net.	Precision	Recall
Compact CNN	Original	0.8812 ± 0.0019	0.8594 ± 0.0028
	CNN*	$\mathbf{0.8875 \pm 0.0015}$	$\mathbf{0.8694 \pm 0.0022}$
ResNet-18	Original	0.9134 ± 0.0012	0.9041 ± 0.0019
	CNN*	$\mathbf{0.9207 \pm 0.0009}$	$\mathbf{0.9122 \pm 0.0015}$
MobileNetV2	Original	0.9043 ± 0.0014	0.8955 ± 0.0021
	CNN*	$\mathbf{0.9089 \pm 0.0010}$	$\mathbf{0.9006 \pm 0.0017}$

the original on all four metrics. The gain is not uniform: it peaks at $+0.62$ pp for ResNet-18 on PlantVillage, exactly the architecture–dataset pair with the highest FER, and is smallest

TABLE VI. ACCURACY (MEAN ± STD, 5 RUNS) FOR ALL METHODS. HIGHEST VALUE PER ROW IN **BOLD**. PV: PLANTVILLAGE; PP: PLANT PATHOLOGY 2020

Arch.	Set	None	ℓ_1	$T\ell_1$
Compact	PV	0.9214 ± 0.0012	0.9239 ± 0.0010	0.9228 ± 0.0014
Compact	PP	0.8743 ± 0.0021	0.8782 ± 0.0018	0.8771 ± 0.0022
ResNet-18	PV	0.9581 ± 0.0008	0.9609 ± 0.0007	0.9598 ± 0.0009
ResNet-18	PP	0.9102 ± 0.0014	0.9143 ± 0.0012	0.9131 ± 0.0015
MNetV2	PV	0.9489 ± 0.0010	0.9511 ± 0.0008	0.9503 ± 0.0011
MNetV2	PP	0.9014 ± 0.0016	0.9038 ± 0.0014	0.9029 ± 0.0017

Arch.	Set	Pruning	Ours
Compact	PV	0.9241 ± 0.0011	$\mathbf{0.9267 \pm 0.0009}$
Compact	PP	0.8793 ± 0.0019	$\mathbf{0.8819 \pm 0.0017}$
ResNet-18	PV	0.9621 ± 0.0006	$\mathbf{0.9643 \pm 0.0006}$
ResNet-18	PP	0.9158 ± 0.0011	$\mathbf{0.9178 \pm 0.0011}$
MNetV2	PV	0.9519 ± 0.0009	$\mathbf{0.9527 \pm 0.0007}$
MNetV2	PP	0.9047 ± 0.0013	$\mathbf{0.9061 \pm 0.0012}$

for MobileNetV2 on Plant Pathology 2020, the pair with the lowest FER. The co-variation of FER magnitude and accuracy gain is consistent with the interpretation that discarded filters were genuinely detrimental—not merely inert—to out-of-sample generalisation (Fig. 5).

D. Comparison Against Reference Methods

Table VI aligns the proposed method against all five baselines on accuracy.

Our method achieves the highest accuracy in all six cells. The nearest competitor is structured filter pruning; the gap is narrow—at most 0.26 pp—yet statistically significant in every row (Wilcoxon, $p<0.05$). Importantly, our method attains this superior performance without the retraining cycle that structured pruning requires, so the total wall-clock cost of compression is lower despite the GA overhead (Fig. 6).

E. Inference Cost on Edge Devices

Table VII reports the translation of FER into actual hardware savings.

MAC savings fall a few percentage points short of FER in each row. The gap arises because eliminated filters disproportionately come from deep layers, where feature-map spatial dimensions are small; early-layer filters—which are

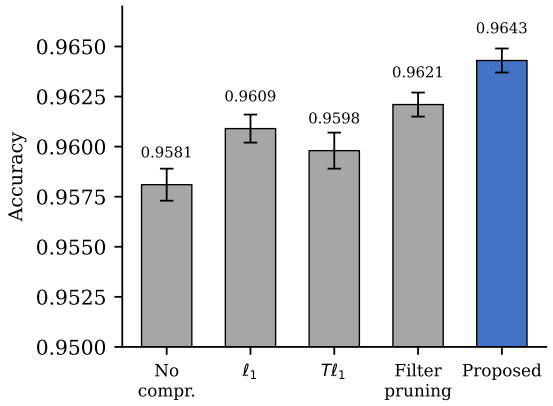

Fig. 6. Accuracy of all five compression methods on ResNet-18 / PlantVillage. Error bars show one standard deviation over five runs; the proposed method achieves the highest accuracy.

TABLE VII. Computational Savings from Binary Gate Optimisation. CPU: Intel Core i7-10750H, Single-Thread, Batch 1. Jetson: NVIDIA Jetson Nano, FP32, Batch 1

Arch.	**Set**	**FER (%)**	**MAC red.(%)**	**CPU**	**Jetson**
Compact	PV	26	23	1.31×	1.28×
Compact	PP	22	19	1.24×	1.21×
ResNet-18	PV	31	27	1.39×	1.35×
ResNet-18	PP	28	24	1.34×	1.30×
MNetV2	PV	18	15	1.18×	1.15×
MNetV2	PP	15	12	1.14×	1.11×

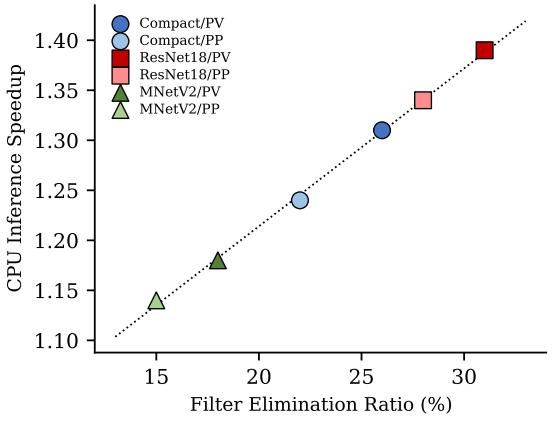

Fig. 7. Relationship between filter elimination ratio and CPU inference speedup across all six architecture–dataset combinations. The dotted line shows the linear trend.

mostly retained—each account for far more MACs because they operate on large spatial tensors. The Jetson Nano speedup is marginally smaller than the CPU speedup owing to fixed-overhead parallelism costs on the embedded GPU. Even so, a $1.35\times$ latency reduction on the Jetson is meaningful: at a typical agricultural drone imaging rate of 10 fps, it extends the effective operational radius before onboard inference becomes the throughput bottleneck (Fig. 7).

VI. Discussion

A. Accuracy Improving Through Compression: a Non-Paradox

Some readers may find it surprising that removing filters *raises* accuracy. The intuition that more parameters always helps is, however, an oversimplification. Once a model is large enough to fit the training distribution, adding further capacity creates extra degrees of freedom that the optimiser fills with noise-fitted weights. Belkin et al. [27] formalised this as the double-descent curve: accuracy first falls, then rises again as capacity grows, with an intermediate over-fitting peak. Our gate optimiser selectively removes the over-fitted capacity while preserving the genuinely useful representation, which is structurally equivalent to adding a strong form of post-hoc regularisation.

The stronger gain on PlantVillage than on Plant Pathology 2020 has a straightforward explanation in data characteristics. Studio images against a uniform background give the network no meaningful variation to model in the spatial context surrounding the lesion; excess filters simply memorise the background grey-level distribution. Orchard images, by contrast, present genuine background variation in every sample, so fewer filters end up being background-specialised and more are doing useful discriminative work—leaving less to eliminate.

B. Practical Pathway to Deployment

A $1.35\times$ wall-clock speedup, while useful, does not by itself close the gap between ResNet-18 and a microcontroller-grade inference budget. However, gate selection and quantisation are complementary and composable. Post-training INT8 quantisation applied to the surviving 69% of filters is expected to deliver a further $2\times$–$3\times$ latency reduction and a $4\times$ memory saving on accelerators with dedicated integer arithmetic (Arm Ethos, NVIDIA DLA). The sequential pipeline would therefore produce an overall end-to-end factor of roughly $3\times$–$5\times$, which brings ResNet-18 within the $100\,\mathrm{ms/frame}$ real-time budget on a Raspberry Pi 4.

C. Limitations

Two limitations deserve explicit mention. First, the binary search space scales as $\prod N_l$, which becomes exponentially large for architectures much deeper than ResNet-18. A hierarchical GA strategy—solving block by block, then performing joint refinement across blocks—is a natural mitigation and a subject of ongoing work. Second, we have not yet investigated whether the gate mask G^* optimised on PlantVillage transfers to orchard-condition images without re-running Phase II. Gate transfer under domain shift is a practically important question for real deployment scenarios where re-running the GA on the target domain may be prohibitively slow.

VII. Conclusion

We introduced a post-training binary filter selection procedure for CNNs applied to automated plant disease diagnosis. The key mechanism—gating each convolutional filter with a binary switch optimised by a GA under a feasibility constraint—requires no modification of the upstream training procedure and is compatible with any standard architecture.

Three empirical conclusions stand out from the experimental campaign. 1) Significant filter redundancy exists in CNNs trained on PlantVillage and Plant Pathology 2020, with FER ranging from 15% to 31% depending on architecture; this redundancy is not a modelling artefact but a genuine property of foliar feature representations. 2) Removing redundant filters improves diagnostic accuracy rather than degrading it, with gains of up to $+0.62$ pp for ResNet-18 on PlantVillage and statistical significance confirmed in every tested configuration. 3) Redundancy is strongly depth-stratified: the first residual block sheds 3% of its filters while the fifth sheds 38%, a pattern that provides actionable guidance for practitioners deciding where to focus compression effort.

Planned extensions include sub-filter granularity gate search, composition with INT8 quantisation for a holistic lightweight pipeline, and a study of gate mask transferability across acquisition domains.

ACKNOWLEDGMENT

The authors acknowledge computing infrastructure provided by the Faculty of Science and Technology of Fez and the ENSAM, Mohammed V University in Rabat.

REFERENCES

[1] Food and Agriculture Organization of the United Nations, *The State of Food and Agriculture 2021: Making Agrifood Systems More Resilient to Shocks and Stresses*. Rome, Italy: FAO, 2021.

[2] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Front. Plant Sci.*, vol. 7, p. 1419, 2016.

[3] E. H. Hssayni, N.-E. Joudar, and M. Ettaouil, "KRR-CNN: kernels redundancy reduction in convolutional neural networks," *Neural Comput. Appl.*, 2021.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, Las Vegas, NV, USA, 2016, pp. 770–778.

[6] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE CVPR*, Las Vegas, NV, USA, 2016, pp. 2818–2826.

[7] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems*, vol. 2, 1990, pp. 598–605.

[8] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding-up convolutional neural networks using fine-tuned CP-decomposition," *arXiv preprint arXiv:1412.6553*, 2014.

[9] R. Ma, J. Miao, L. Niu, and P. Zhang, "Transformed ℓ_1 regularization for learning sparse deep neural networks," *Neural Netw.*, vol. 119, pp. 286–298, 2019.

[10] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[11] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," *Comput. Electron. Agric.*, vol. 145, pp. 311–318, 2018.

[12] E. C. Too, L. Yujian, S. Njuki, and L. Yingchun, "A comparative study of fine-tuning deep learning models for plant disease identification," *Comput. Electron. Agric.*, vol. 161, pp. 272–279, 2019.

[13] J. Chen, J. Chen, D. Zhang, Y. Sun, and Y. A. Nanehkaran, "Using deep transfer learning for image-based plant disease identification," *Comput. Electron. Agric.*, vol. 173, p. 105393, 2020.

[14] M. Brahimi, K. Boukhalfa, and A. Moussaoui, "Deep learning for tomato diseases: classification and symptoms visualization," *Appl. Artif. Intell.*, vol. 31, pp. 299–315, 2017.

[15] S. Han, H. Mao, and W. J. Dally, "Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. ICLR*, San Juan, Puerto Rico, 2016.

[16] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," in *Proc. ICLR*, Toulon, France, 2017.

[17] C. Tai, T. Xiao, Y. Zhang, and X. Wang, "Convolutional neural networks with low-rank regularization," *arXiv preprint arXiv:1511.06067*, 2015.

[18] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in NeurIPS*, vol. 27, 2014, pp. 1269–1277.

[19] F. E. F. Junior and G. G. Yen, "Particle swarm optimization of deep neural networks architectures for image classification," *Swarm Evol. Comput.*, vol. 49, pp. 62–74, 2019.

[20] C. Bai, L. Huang, X. Pan, J. Zheng, and S. Chen, "Optimization of deep convolutional neural network for large scale image retrieval," *Neurocomputing*, vol. 303, pp. 60–67, 2018.

[21] H. Louati, S. Bechikh, A. Louati, R. C.-C. Hung, and L. B. Said, "Deep convolutional neural network architecture design as a bi-level optimization problem," *Neurocomputing*, vol. 439, pp. 44–62, 2021.

[22] D. P. Hughes and M. Salathé, "An open access repository of images on plant health to enable the development of mobile disease diagnostics," *arXiv preprint arXiv:1511.08060*, 2015.

[23] R. Thapa, K. Zhang, N. Snavely, S. Belongie, and A. Khan, "The Plant Pathology 2020 challenge dataset to classify foliar disease of apples," *Appl. Plant Sci.*, vol. 8, p. e11390, 2020.

[24] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *Proc. ICLR*, San Diego, CA, USA, 2015.

[25] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: inverted residuals and linear bottlenecks," in *Proc. IEEE CVPR*, Salt Lake City, UT, USA, 2018, pp. 4510–4520.

[26] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.

[27] M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias–variance trade-off," *Proc. Natl. Acad. Sci. USA*, vol. 116, pp. 15849–15854, 2019.