

Tomato leaf disease classification and parameter optimisation using a hybrid deep learning framework

Sachin Jain^{a,b}, Chin-Shiuh Shieh^a, Vishal Jain^c, Rajesh Prasad^{b,d}, Muhammad Musa Liman^{d,*}

^aResearch Institute of IoT Cybersecurity, Department of Electronic Engineering, National Kaohsiung University of Science and Technology, Taiwan

^bDepartment of Computer Science and Engineering, Ajay Kumar Garg Engineering College, Ghaziabad, Uttar Pradesh, India

^cDepartment of Computer Science & Engineering, School of Engineering & Technology, Vivekananda Institute of Professional Studies–Technical Campus, New Delhi, India

^dDepartment of Computer Science, African University of Science and Technology, Abuja, Nigeria

Abstract

Agriculture underpins national economies, and tomatoes are among the most widely grown vegetables, yet tomato crops are highly susceptible to leaf diseases that reduce yields and farm incomes. Single-backbone convolutional neural networks and transfer-learning approaches often rely on generic features and struggle to separate visually similar diseases. This study proposes a hybrid deep learning framework in which two frozen, ImageNet-pretrained convolutional neural networks (CNNs), EfficientNetB0 and DenseNet121, extract complementary features that are concatenated and passed to a lightweight classifier. The learning rate, number of dense units, and dropout rate of the classifier were selected with a customised, population-based honey badger optimisation search. Images from a public ten-class tomato leaf dataset were resized and normalised, and a balanced subset of 2,000 images was used for training and tuning. Over 35 repeated training runs on a fixed training/validation split, the model achieved a mean validation accuracy of 82.1% (standard deviation 1.3 percentage points). The final model, trained for 15 epochs, reached 87.0% on the validation split and 87.5% (95% confidence interval 85.3–89.4%) on 1,000 independent images that were never used for training or tuning, with a macro-averaged F1-score of 0.875. Viral and bacterial diseases were recognised most reliably, whereas early blight and target spot remained the most difficult classes. The results show that frozen dual-backbone feature fusion provides a compact baseline for tomato leaf disease classification, although the benefit of the hyperparameter search over default settings was not established.

DOI: [10.46481/asr.2026.5.3.616](https://doi.org/10.46481/asr.2026.5.3.616)

Keywords: Tomato leaf disease, Hybrid deep learning, Hyperparameter optimisation, Honey badger optimisation algorithm, Feature fusion

Article History:

Received: 30 July 2026

Received in revised form: 07 September 2026

Accepted for publication: 08 October 2026

Available online: 22 October 2026

© 2026 The Author(s). Published by the Nigerian Society of Physical Sciences under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

1. Introduction

Agriculture has supported human civilization by growing food and raising cattle. A major issue today is food insecurity, which is sometimes linked to plant diseases [1]. Disease detection is crucial in agriculture to anticipate and prevent crop losses, making

*Corresponding Author Tel. No.: +234-704-959-3359.

Email address: mliman@student.aust.edu.ng (Muhammad Musa Liman)

proactive plant health monitoring and prediction essential. Automatic leaf disease identification has become popular due to the necessity to manage and monitor large agricultural fields [2]. Plant diseases limit agricultural yield and cause food shortages. According to the FAO, plant pests and diseases destroy up to 40% of global crop production. Given the economic and environmental impacts of contemporary agriculture, early detection of tomato plant diseases is crucial worldwide.

Many leaf diseases affect tomato plants every year. Farmer well-being, income, and savings are all impacted by decreased harvest yields. For optimal growth and crop productivity, accurate detection of tomato leaf diseases is necessary [3, 4]. Diseases can cause tomato harvests to be of poor quality, which has repercussions for both human health and economic stability [5]. Experts in botany used to have to examine tomato leaves for signs of disease physically. This approach is notoriously tedious and prone to errors [6]. Such inaccurate diagnoses can significantly impact tomato fruit quality. Much work has been done in plant disease identification and classification over the past 30 to 40 years. Good crop yield and the avoidance of losses depend on prompt, precise detection [7]. Image recognition technology is increasingly used to detect plant diseases, driven by advances in digital and visual components. Accurate and timely identification of leaf diseases therefore helps protect yields and farm incomes.

Finding plant diseases in images has recently become possible using methods such as computer vision, machine learning (ML), and deep learning (DL) [8–10]. Without the assistance of domain experts, DL methods modelled after the human brain can automatically learn to distinguish between different picture features. Advancements in DL models have successfully addressed the issues with conventional ML methods. Chouhan *et al.* [10] reported that various deep learning techniques, such as recurrent neural networks (RNNs), long short-term memory networks (LSTMs), and convolutional neural networks (CNNs), have been employed to detect diseases on plant leaves [11].

Because it can analyse patterns in a manner comparable to humans, DL excels at classifying agricultural plant diseases [12]. Deep learning enables models to identify various diseases in plant images by learning subtle details and patterns without human intervention in defining these features in advance, leading to quicker, more accurate detection. Computer vision tasks, such as picture recognition, have recently seen remarkable advancements thanks to deep learning [3]. CNNs, a type of DL, are widely used for plant disease classification [13]. Overall, the DL technique has improved classification and automated plant leaf disease categorisation, saving time and labour [14]. Ayode *et al.* [15] further showed that systematic hyperparameter optimisation materially affects the performance of machine-learning-based leaf disease detection in maize, which motivates the optimisation-based approach adopted in this study.

Tomatoes are vital to global food security; therefore, detecting leaf diseases is crucial to public health. This detection is difficult because disease symptoms vary, field conditions change, and detection methods do not always work.

This paper aimed to create an effective model for detecting tomato leaf disease. The method involves several stages. First, input leaf images undergo preprocessing. In the second step, a hybrid of two pre-trained models, optimised using the customised honey badger optimisation algorithm (CHBOA), performs multiclass plant disease classification. This model classifies tomato leaves as early blight, healthy, bacterial spot, two-spotted spider mite, leaf mould, septoria leaf spot, tomato mosaic virus, late blight, tomato yellow leaf curl virus, and target spot.

Anim-Ayeko *et al.* [9] developed a ResNet-9 model for detecting blight disease in tomatoes and potatoes using leaf images. The model considered the leaf shape, the overall green area, and the disease-affected area of the leaf to predict the disease. Even though the model was capable of detecting early and late blight disease and the absence of disease using leaf images, it failed to improve when more environmental data and agricultural management were included [9]. Al-gaashani *et al.* [4] proposed a simple transfer-learning and feature-concatenation approach for classifying tomato leaf disease, in which a conventional classifier was incorporated to improve performance and reduce training and testing time.

However, the model was ineffective at exploring and implementing new transfer learning approaches involving feature extraction from multi-layered pre-trained models to identify the most efficient layers for optimal feature selection. Li *et al.* [16] introduced a GD-Attention mechanism that guides the model to extract features from the pixel-value distribution, helping to locate the diseased area of the leaf. Although this method achieved robustness and higher accuracy with a minimum number of parameters, it failed to gather more complex scenario data for optimisation. Islam *et al.* [5] devised an attention-based dilated CNN with logistic regression (ADCLR) for leaf disease detection (see also [17]). In this method, a Conditional Generative Adversarial Network (CGAN) was used to produce a synthetic image from a preprocessed image, with bilateral filtering and Otsu segmentation applied. This approach yielded the best prediction results when handling imbalanced and noisy data.

However, this method was unsuccessful in reducing time and space complexity. Bajpai and Tyagi [18] introduced the Enhancement Segmentation-CNN (ES-CNN) model to detect leaf disease at an early stage. In addition to the ES-CNN model, mean filtering and CLAHE are used to enhance the image, and k-means clustering is used for image segmentation.

Even though this model improved analytical capabilities, it failed to extend disease identification to other parts of the plant, such as stems, flowers, and fruits. Albahli and Nawaz [12] used a DenseNet-77-based CornerNet model to classify abnormalities in tomato plant leaves; the model achieved efficiency by avoiding overfitting, but it had difficulty with images showing large changes in viewing angle. Albattah *et al.* [1] developed an improved EfficientNet-based framework for the detection and classification of plant leaf diseases. Khan *et al.* [19] combined the scale-invariant feature transform (SIFT) with the grey-level co-occurrence matrix (GLCM) for feature extraction, followed by a support vector machine (SVM) for classification.

While ResNet-9, transfer learning, GD-Attention, and standard hybrid architectures are promising, they have significant draw-

Table 1: Review of literature.

Ref	Method	Merit	Demerit
[1]	EfficientNetV2	This method provided a low-cost solution for classifying plant leaf diseases in real-world settings.	It failed to extend its approach to other parts of plants, such as stems, and proved ineffective when evaluated on complex databases.
[4]	Transfer Learning and a concatenated-based approach	The model's training and testing times were low.	It failed to recognize the most efficient layers for optimal feature selection.
[5]	ADCLR	This model produced the best prediction result by handling imbalanced and noisy data.	This method was unsuccessful in reducing complexity.
[9]	Resnet-9	It can detect early and late blight disease, as well as the absence of disease, using leaf images.	It did not consider environmental data.
[12]	DenseNet-77-based CornerNet	The model's efficiency was high, as it avoided overfitting.	Its detection performance degraded on images with large changes in viewing angle.
[16]	GD-Attention mechanism	It achieved robustness and higher accuracy with a minimum number of parameters.	It failed to gather more complex data for optimization.
[18]	ES-CNN	This model achieved enhanced analytical capabilities.	It failed to expand disease identification to other parts of the plant, such as stems, flowers, and fruits.
[19]	SIFT + GLCM with SVM	It helps in precise and timely disease management.	It considered only texture features for classification.

backs when it comes to plant disease categorisation. ResNet-9 achieves above-average accuracy when learning hierarchical features, but it is not very deep and has trouble capturing fine-grained patterns of disease. Although transfer learning utilises pre-trained weights to enhance performance, the domain-specific variances in tomato leaf disease make it less adaptable due to its generic features. GD-Attention highlights diseased regions but plateaus because of ineffective parameter adjustment. Conventional hybrid models have poor accuracy due to their instability and propensity for overfitting. Table 1 provides an overview of the current literature, including its benefits and drawbacks.

The key contributions of this work are listed below:

- Created a hybrid architecture using transfer learning and feature-level concatenation of two backbone networks to collect generic and domain-specific tomato leaf disease features.
- Implemented a population-based CHBOA search for the learning rate, dense units, and dropout rate of the classifier, and report its full settings and selected values.
- Quantified run-to-run variability over 35 repeated training runs and evaluated the final model on an independent 1,000-image test set that was never used for training or tuning.

The subsequent sections of this paper are arranged as follows. Section 2 provides a detailed description of our proposed CHBOA_Hybrid. Section 3 presents our experimental results, Section 4 discusses them, and Section 5 concludes the work.

2. Materials and methods

2.1. Collection of dataset

The tomato leaf disease detection dataset [20] contains 11,000 RGB images of tomato leaves in 10 classes. It is distributed as a training folder of 10,000 images (1,000 per class) and a validation folder of 1,000 images (100 per class), so the source dataset is exactly balanced, with 1,100 images per class. Every CHBOA fitness evaluation requires training the full hybrid model, and the study required 25 such evaluations plus 36 further training runs. Using all 11,000 images for these experiments was therefore computationally infeasible within the available Google Colaboratory GPU budget. A class-balanced subset of 200 images per class (2,000 images in total) was drawn from the training folder by stratified random sampling without replacement (random_state = 42).

Because the source dataset is itself exactly balanced, drawing an equal number of images per class preserves its uniform class distribution. No images were excluded, and no explicit duplicate screening was performed. Because the dataset is derived from

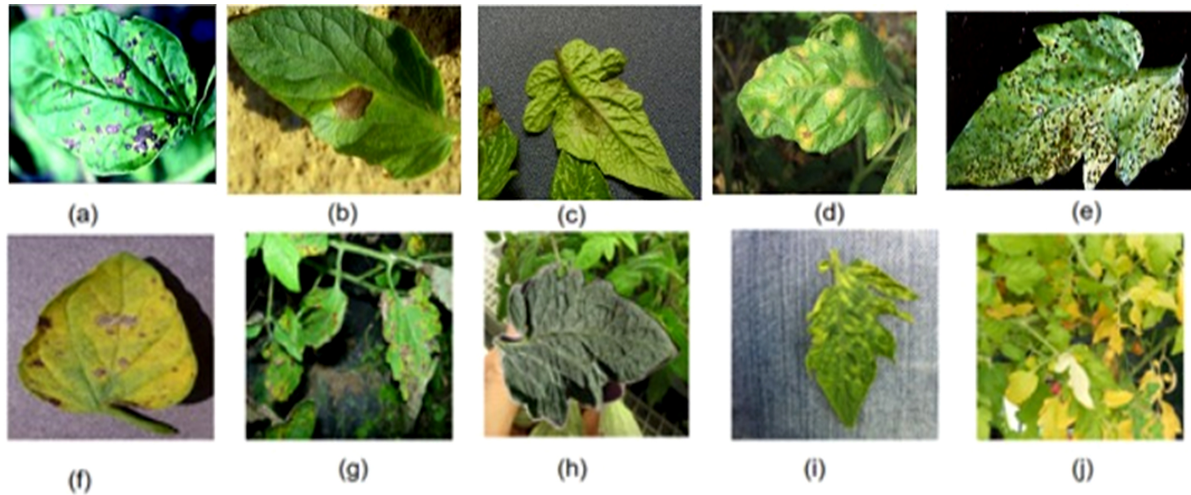


Figure 1: Sample images of the ten tomato leaf classes.

PlantVillage and may contain near-identical images of the same leaf, some near-duplicates may fall on both sides of the training/validation split, which could inflate the reported validation accuracies. The 2,000-image pool was split per class into 1,600 training images (160 per class) and 400 validation images (40 per class), using the Keras ImageDataGenerator validation_split = 0.2. This option deterministically assigns the last 20% of the files in each class folder to validation. The dataset's separate 1,000-image validation folder (100 per class) was not used for training, hyperparameter search, or model selection, and serves as the independent test set (Section 3).

The list of sampled file names is available from the corresponding author on request. Figure 1 presents the 10 distinct classes of tomato leaf images considered in this study.

Figure panel classes: (a) Tomato___Bacterial_spot

(b) Tomato___Early_blight

(c) Tomato___Late_blight

(d) Tomato___Leaf_Mold

(e) Tomato___Septoria_leaf_spot

(f) Tomato___Spider_mites_Two-spotted_spider_mite

(g) Tomato___Target_Spot

(h) Tomato___Tomato_Yellow_Leaf_Curl_Virus

(i) Tomato___Tomato_mosaic_virus

(j) Tomato___healthy

2.2. Method and algorithm

Tomatoes are the most widely consumed vegetables worldwide. Moreover, they provide significant contributions to human health and food security. Therefore, early prediction and classification of tomato leaf diseases play a vital role in the production of food quality and quantity. Early detection of disease helps minimise the extensive impact of agricultural losses and ensure stable supply chains. Thus, in this research, an innovative model, CHBOA_Hybrid, is employed to detect diseases in tomato leaves. Initially, each input image from the dataset [20] is resized to $224 \times 224 \times 3$ pixels and its pixel values are rescaled to [0, 1] [21]; no data augmentation was applied. After preprocessing, plant leaf disease classification is performed using two powerful CNN architectures, EfficientNetB0 [22] and DenseNet121 [23].

The hyperparameter search of CHBOA_Hybrid is inspired by the Honey Badger Algorithm (HBA) [24] and the Skill Optimization Algorithm (SOA) [25] (Section 2.3). The trained hybrid model classifies the input image into 10 leaf disease classes: early blight, healthy, bacterial spot, two-spotted spider mite, leaf mould, septoria leaf spot, tomato mosaic virus, late blight, tomato yellow leaf curl virus, and target spot. The block diagram of CHBOA_Hybrid for detecting leaf disease in tomatoes is portrayed in Figure 2.

Algorithm 1. Hybrid Model Construction

Input:

img_size: Input image size ($224 \times 224 \times 3$)

dense_units: Number of units in the fully connected layer

dropout_rate: Dropout rate

lr: Learning rate
 10 Classes: Number of output classes
 Output:
 M: Compiled hybrid model
 Perform
 For the input layer Do
 Define an input tensor of shape = (224, 224, 3).
 For backbone feature extractors, Do
 Load EfficientNetB0 pretrained on ImageNet, excluding the top layer, and set trainable = False.
 Load DenseNet121 pretrained on ImageNet, excluding the top layer, and set trainable = False.
 For feature aggregation, Do
 Global average pooling on EfficientNetB0 output $\rightarrow x_1$
 Apply global average pooling on DenseNet121 output $\rightarrow x_2$
 Concatenate pooled features: combined = $[x_1, x_2]$
 For fully connected layers, Do
 Apply a dense layer with dense_units and ReLU activation
 Apply Dropout with dropout_rate
 For the output layer Do
 Apply a Dense layer with len(class_labels) units and Softmax activation
 For compiling the model Do
 Optimizer: Adam (lr)
 Loss: Categorical Crossentropy
 Metrics: Accuracy
 Return
 Compiled model M

Dataset D is defined in Eq. (1):

$$D = \{(x_i, y_i)\}_{i=1}^N, \quad (1)$$

where $x_i \in \mathbb{R}^d$ is the input feature vector (tomato leaf image features), $y_i \in \{1, 2, \dots, C\}$ is the class label (disease category), N is the total number of samples, and C is the number of classes.

The hybrid model extracts features from EfficientNetB0, shown in Eq. (2), and DenseNet121, shown in Eq. (3):

$$F_E = f_E(x; \theta_E), \quad (2)$$

$$F_D = f_D(x; \theta_D), \quad (3)$$

where f_E is the EfficientNetB0 feature extractor and f_D is the DenseNet121 feature extractor.

The two pooled feature vectors are fused by concatenation, as defined in Eq. (4):

$$F_H = F_E \parallel F_D, \quad (4)$$

where $\parallel$ denotes concatenation of the 1,280-dimensional EfficientNetB0 vector and the 1,024-dimensional DenseNet121 vector, giving a 2,304-dimensional fused vector. Concatenation is the only fusion operation used in this study (consistent with Algorithm 1 and Figure 3); no weighted fusion coefficients are defined, learned, or optimised.

The final prediction is defined in Eq. (5):

$$\hat{y} = \text{Softmax}(WF_H + b), \quad (5)$$

where WF_H is the weight matrix and b is the bias vector.

The loss function (multiclass cross-entropy) is defined in Eq. (6):

$$L = - \sum_{i=1}^N \sum_{c=1}^C y_{ic} \log \hat{y}_{ic}. \quad (6)$$

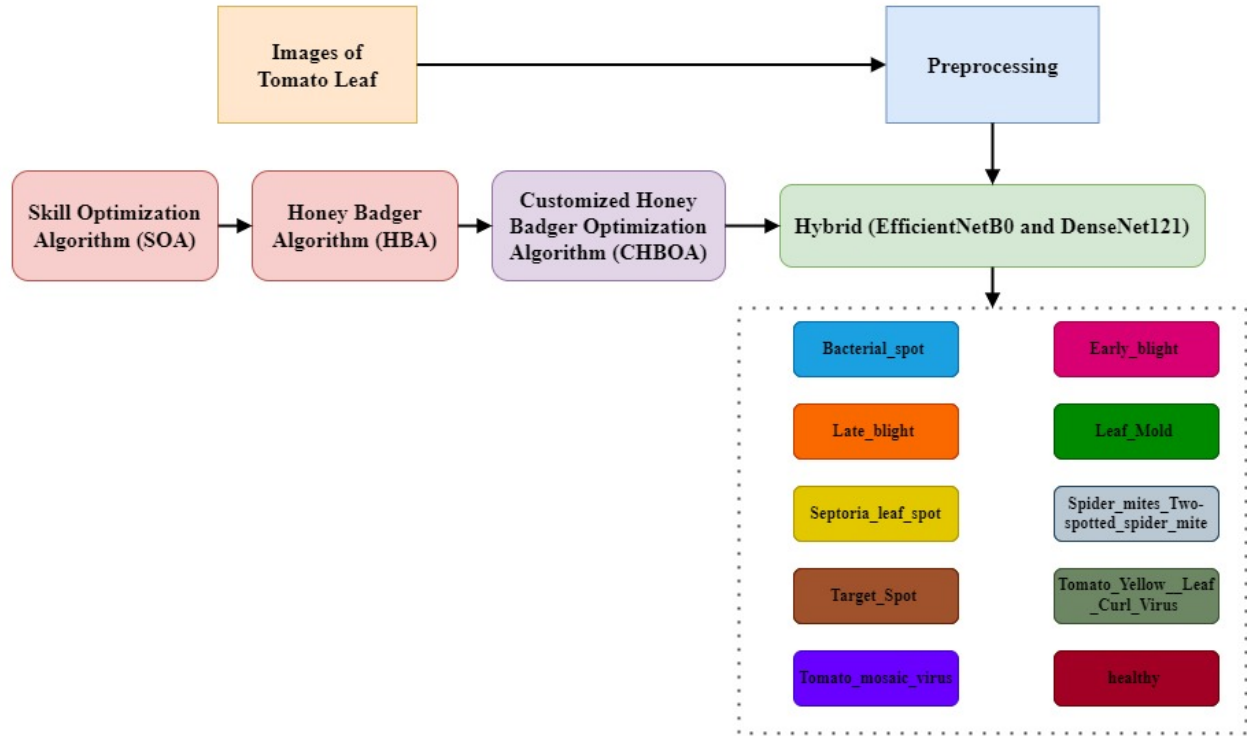


Figure 2: Block diagram of the devised CHBOA_Hybrid for leaf disease classification in tomato.

The proposed hybrid deep learning model combines the strengths of two powerful CNN architectures, EfficientNetB0 and DenseNet121, both pre-trained on the ImageNet dataset, shown in Figure 3. Instead of using them in isolation, the model integrates their feature-extraction capabilities by removing their top classification layers and freezing their weights to retain the rich, pre-trained knowledge. Input images of size (224, 224, 3) are processed through both networks, and global average pooling is applied to condense the spatial features into compact, informative representations. EfficientNetB0 [22] and DenseNet121 [23] were selected over other common backbones (e.g., ResNet, VGG, and MobileNet) because they combine strong ImageNet transfer-learning accuracy with a relatively small parameter count (EfficientNetB0's compound scaling and DenseNet121's dense feature-reuse connections), which keeps the number of trainable parameters low when both backbones are frozen and only lightweight fusion/classification layers are trained; the two architectures are also complementary, as EfficientNetB0 emphasises efficient depth/width/resolution scaling while DenseNet121 emphasises fine-grained feature reuse across layers. To reduce retraining, both models are started with ImageNet weights and frozen. Each backbone extracts rich hierarchical features, which Global Average Pooling layers then refine to minimise dimensionality while preserving vital information.

2.3. CHBOA hyperparameter search

Each agent j holds a candidate hyperparameter vector, as defined in Eq. (7):

$$P_j = (lr_j, du_j, dr_j, bs_j), \quad (7)$$

where lr_j is the learning rate, du_j the number of dense units, dr_j the dropout rate, and bs_j the batch-size variable. Each component is initialised uniformly within the bounds of Table 2. The fitness of an agent is its validation accuracy, as defined in Eq. (8), and is maximised:

$$F(P_j) = \text{Acc}_{\text{val}}(M(P_j)), \quad (8)$$

where $M(P_j)$ is the hybrid model of Algorithm 1 built with lr_j , $[du_j]$, and dr_j , trained for 2 epochs on the 1,600-image training partition and evaluated on the 400-image validation partition. Maximising F is equivalent to minimising the classification error $1 - \text{Acc}_{\text{val}}$; no loss or model-complexity term is used. After all agents are evaluated, each coordinate d of each agent moves toward the best agent found so far, P^* , as defined in Eq. (9):

$$P_{j,d} \leftarrow \text{clip}(P_{j,d} + r \cdot (P_d^* - P_{j,d}), lb_d, ub_d), \quad r \sim U(-1, 1). \quad (9)$$

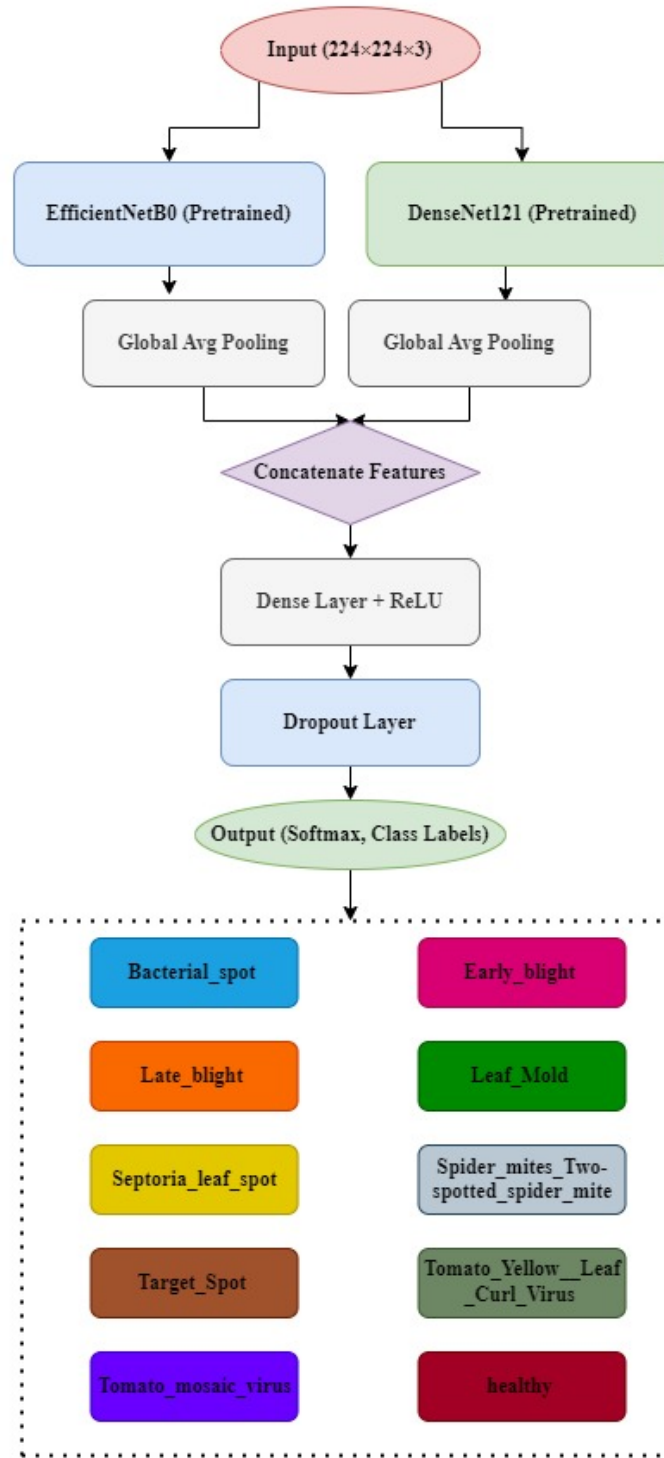


Figure 3: Architecture of the hybrid EfficientNetB0–DenseNet121 model with CHBOA hyperparameter tuning.

where r is drawn independently for every agent and coordinate, and lb_d and ub_d are the bounds of Table 2 (boundary handling by clipping). Values of r close to 1 pull an agent toward the best solution (exploitation), while negative or small values of r move it away from or around it (exploration). This attraction toward the incumbent best follows the movement idea of HBA [24] and SOA [25]. The full digging- and honey-phase operators of HBA and a separate SOA phase were not implemented, so the search is a simplified, population-based variant.

The two discrete variables are handled as follows. Dense units are truncated to an integer ($\lfloor du \rfloor$) when the model is built. The

batch-size variable is truncated to an integer, but in the implementation it did not change the mini-batch size, which was fixed at 32 by the data generator. It only set the number of training steps per epoch to $\lfloor 50/\text{bs} \rfloor$ (the 1,600 training images form 50 batches of 32). When this value was 0 ($\text{bs} > 50$), Keras trained a full epoch of 50 steps; otherwise each epoch used only 1–3 batches. With 2 epochs per fitness evaluation, candidates with $\text{bs} \leq 50$ therefore received only $2 \sim 6$ weight updates, whereas candidates with $\text{bs} > 50$ received 100. This behaviour is discussed in Section 4.

Eqs. (1)–(6) are standard formulations for supervised deep-learning classification (dataset definition, feature extraction, softmax output, and categorical cross-entropy loss) and are not attributed to a single external source. Algorithm 2 gives the complete procedure used to generate the reported results.

Algorithm 2. CHBOA hyperparameter search (implemented procedure)

Input: bounds S (Table 2); $N_a = 5$ agents; $T = 5$ iterations; $E_{\text{fit}} = 2$ epochs; training partition D_{train} (1,600 images) and validation partition D_{val} (400 images)

Output: P^* (best hyperparameter set); Acc^* (best validation accuracy)

1. For $j = 1, \dots, N_a$: initialise each component of P_j uniformly within S . Set $P^* = \text{None}$, $\text{Acc}^* = -1$.

2. For $t = 1, \dots, T$ do

a. For each agent j : build M with Algorithm 1 using $(l_r, [d_{uj}], d_r)$; train for E_{fit} epochs on D_{train} with mini-batch 32 and $\text{steps_per_epoch} = \lfloor 50/\text{bs}_j \rfloor$ (a full epoch if 0); record $F_j = \text{Acc}_{\text{val}}$ on D_{val} .

b. For each j : if $F_j > \text{Acc}^*$, set $\text{Acc}^* \leftarrow F_j$ and $P^* \leftarrow P_j$.

c. For each agent j and each dimension d : draw $r \sim U(-1, 1)$ and update $P_{j,d}$ with Eq. (9), including clipping to S .

End for

3. Terminate after T iterations (fixed budget of $N_a \times T = 25$ fitness evaluations; no early stopping).

4. Return P^* and Acc^* .

2.4. Repeated-run evaluation and experimental settings

Algorithm 3. Repeated-run evaluation of the hybrid model

Input: M — hybrid model constructor (Algorithm 1); $P^* = \{l_r, d_u, d_r\}$ from Algorithm 2; D_{train} (1,600 images), D_{val} (400 images); $R \in \{5, 10, 20\}$ numbers of runs; $E = 8$ epochs

Output: mean accuracy Acc_R and the individual run accuracies for $R = 5, 10, 20$

For each R in $\{5, 10, 20\}$ do

For run $r = 1, \dots, R$:

Build a new model $M_r = \text{create_hybrid_model}(224, [d_u], d_r, l_r)$ with a fresh random initialisation of the trainable layers.

Train M_r on D_{train} for E epochs (mini-batch 32, shuffled).

$\text{Acc}_r = \text{Accuracy}(M_r, D_{\text{val}})$; store Acc_r .

$\text{Acc}_R = (1/R) \sum \text{Acc}_r$.

Return $\text{Acc}_5, \text{Acc}_{10}, \text{Acc}_{20}$ and all 35 run accuracies.

Algorithm 3 describes exactly the procedure that produced Table 5. In the original code, a KFold ($n_splits = R$, $\text{shuffle} = \text{True}$, $\text{random_state} = 42$) splitter was created, but its fold indices were never applied to the data. As a result, every run trained on the same 1,600 images and was evaluated on the same 400 images. They are 35 repeated training runs on one fixed split, and they measure run-to-run variability caused by random initialisation and data shuffling. They are reported as such here (Table 5 and Table 6).

CHBOA was run with $N_a = 5$ agents for $T = 5$ iterations (25 fitness evaluations), with 2 training epochs per evaluation and a stopping rule of a fixed iteration budget. The random draws of the search were not seeded. The repeated runs (Algorithm 3) used 8 epochs each. The final model was trained for 15 epochs with P^* on D_{train} , saved, and evaluated once on the independent test set. All runs used the Adam optimiser, categorical cross-entropy, a mini-batch size of 32, and 224×224 inputs rescaled to $[0, 1]$. All experiments were executed in the Google Colaboratory GPU runtime with Python 3.12, TensorFlow 2.19.0, Keras 3.10.0, and scikit-learn 1.6.1; the specific GPU model and the total wall-clock time of the search were not logged. The search space, settings, and selected values are listed in Table 2.

3. Results

3.1. CHBOA results

The CHBOA search (Algorithm 2) evaluated 25 candidate configurations. The best validation accuracy, $\text{Acc}^* = 0.750$ (75.0% after 2 epochs), was reached in the first iteration and was not exceeded in the four subsequent iterations. The selected configuration P^* is listed in Table 2.

Table 2: Hyperparameter search space, CHBOA settings, and selected values.

Hyperparameter / setting	Range	CHBOA-selected / used value
Learning rate (lr)	$[1 \times 10^{-5}, 1 \times 10^{-3}]$ (uniform)	6.63×10^{-4}
Dense units	[128, 512]	404
Dropout rate (dr)	[0.2, 0.6]	0.571
Batch-size variable (bs)	[16, 64]	58 (sets steps per epoch = $\lfloor 50/\lfloor bs \rfloor \rfloor$ only, full 50-step epoch when $bs > 50$; mini-batch fixed at 32)
Number of agents Na	—	5
Number of iterations T	—	5 (25 fitness evaluations)
Epochs per fitness evaluation	—	2
Fitness	—	Validation accuracy on the 400-image split (maximised)
Stopping rule	—	Fixed budget of T iterations
Best fitness Acc*	—	0.750 (reached in iteration 1)
Epochs: repeated runs / final model	—	8/15
Software / hardware	—	Python 3.12, TensorFlow 2.19.0, Keras 3.10.0; Google Colaboratory GPU runtime

Table 3: Performance metrics.

Metrics	Formula
Accuracy	$(TP + TN) / (TP + TN + FP + FN)$
Precision	$TP / (TP + FP)$
F1 Score	$2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$
Recall	$TP / (TP + FN)$

A number of metrics are used to assess the proposed classification model, including recall, accuracy, precision, and F1-score. Table 3 contains the performance evaluation formulas. For the 10-class problem, TP, TN, FP, and FN are computed per class using a one-vs-rest scheme; macro-average precision, recall, and F1-score are the unweighted mean of the per-class values, while weighted-average metrics weight each class's score by its support (number of true instances of that class), consistent with the standard scikit-learn classification_report convention.

Figure 4 and Table 4 are for the final 15-epoch model evaluated on the 400-image validation split (40 images per class). This split was also used to compute the CHBOA fitness and the repeated-run accuracies. Results for the same model on the independent test set are given in Table 7. With the majority of predictions along the diagonal, Figure 4's confusion matrix shows that the tomato disease classification model performs effectively. Leaf Mold, Tomato Mosaic Virus, and Healthy are classified with high accuracy, and Late Blight, Spider Mites, and Bacterial Spot are also generally well predicted. The main misclassifications occur between visually similar diseases, such as Early Blight with Late Blight and Target Spot with Late Blight.

The proposed model's performance is shown in Table 4 and Figure 5. The model performs exceptionally well for Tomato_Yellow_Leaf_Curl_Virus (Precision = 0.972, Recall = 0.875, F1-score = 0.921) and Tomato_mosaic_virus (F1-score = 0.951), indicating strong detection and minimal misclassification. Bacterial spot and Healthy leaves likewise have high F1-scores (0.884 and 0.937, respectively), indicating accurate discrimination.

Early blight (F1-score = 0.720) and Target Spot (F1-score = 0.780) show more modest performance, most likely because of their visual similarity to other fungal diseases (the classes are exactly balanced, so class imbalance is not a factor). A consistent and balanced performance across categories is indicated by F1-scores above 0.85 in the majority of classes.

With an accuracy of 87.0%, roughly 87% of leaf samples in this validation split are correctly classified. The macro average (0.873 precision, 0.870 recall, and 0.870 F1-score) gives equal weight to each class. The weighted average is identical to the macro average here because every class has equal support.

While the suggested model is good at detecting bacterial spots and viral infections, it could be improved in distinguishing between fungal diseases that look very similar, such as Early blight and Target Spot.

Table 5 summarises the 35 repeated training runs (Algorithm 3); the individual run accuracies are listed in Table 6. The mean validation accuracies were 83.25% (5 runs), 81.95% (10 runs), and 81.94% (20 runs), with standard deviations of 1.0 – 1.4 percentage points. Pooled over all 35 runs, the mean was 82.13% (SD 1.31; 95% CI 81.68–82.58%; range 79.75–85.00%). Because the three groups use an identical procedure, their 95% confidence intervals overlap and the differences between them reflect only random run-to-run variation. Figure 6 compares these values with the final model.

The hybrid architecture has 12,022,347 parameters, 935,270 of which are trainable. The design choice was to freeze the pre-

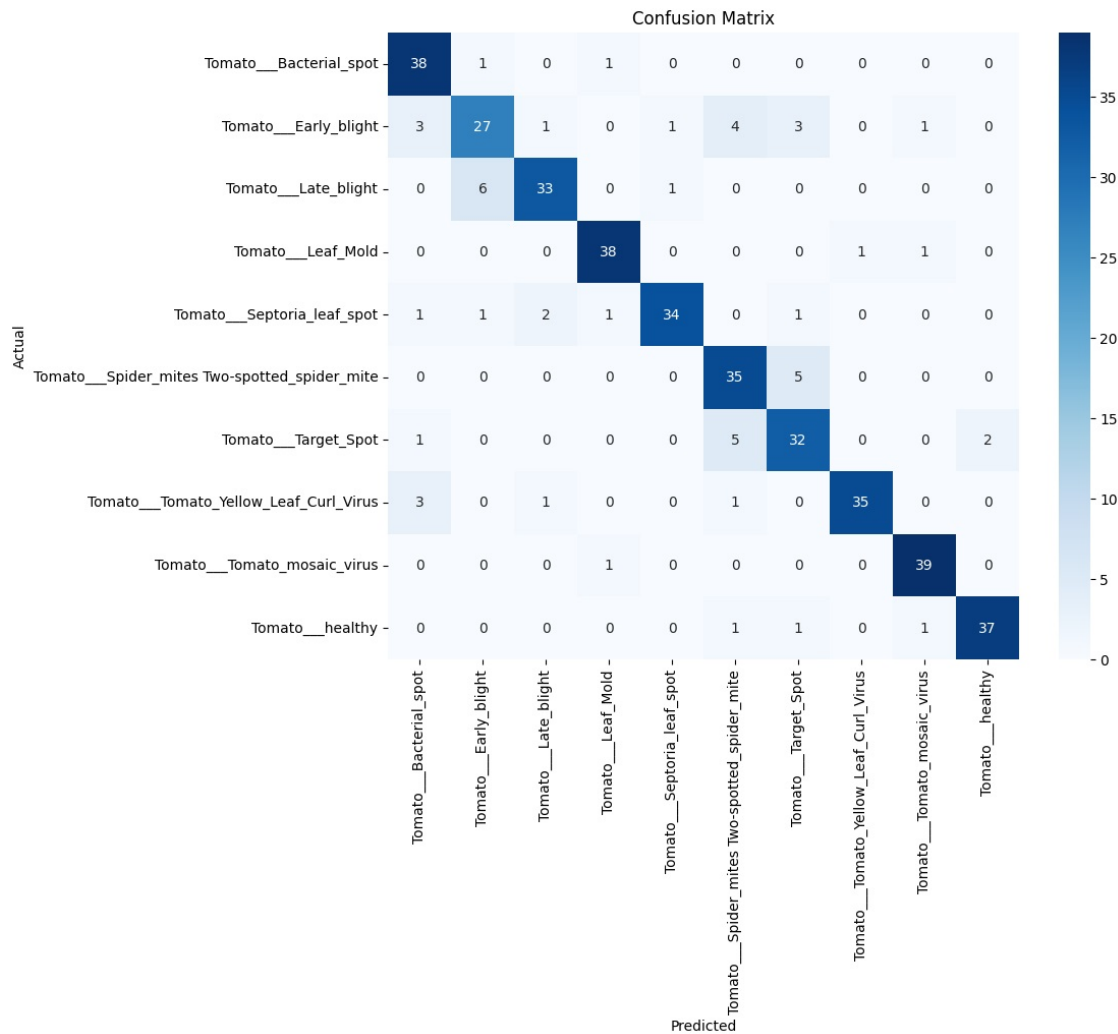


Figure 4: Confusion matrix of the final model on the 400-image validation split.

Table 4: Performance results.

Different Classes	Precision	Recall	F1-score
Tomato__Bacterial_spot	82.6	95.0	88.4
Tomato__Early_blight	77.1	67.5	72.0
Tomato__Late_blight	89.2	82.5	85.7
Tomato__Leaf_Mold	92.7	95.0	93.8
Tomato__Septoria_leaf_spot	94.4	85.0	89.5
Tomato__Spider_mites_Two-spotted_spider_mite	76.1	87.5	81.4
Tomato__Target_Spot	76.2	80.0	78.0
Tomato__Tomato_Yellow_Leaf_Curl_Virus	97.2	87.5	92.1
Tomato__Tomato_mosaic_virus	92.9	97.5	95.1
Tomato__healthy	94.9	92.5	93.7
Accuracy	87.0	87.0	87.0
Macro Average	87.3	87.0	87.0
Weighted Average	87.3	87.0	87.0

trained layers of EfficientNetB0 and DenseNet121 and train only the fully connected layers. The 11,087,075 non-trainable parameters retain ImageNet-pretrained information, saving training time and mitigating overfitting. Of the trainable parameters, 931,220 belong to the 404-unit dense layer ($2,304 \times 404 + 404$) and 4,050 to the 10-class output layer ($404 \times 10 + 10$). The two optimizer parameters reported by the Keras model summary (12 bytes in total) are the Adam optimizer's scalar state variables, namely the iteration counter

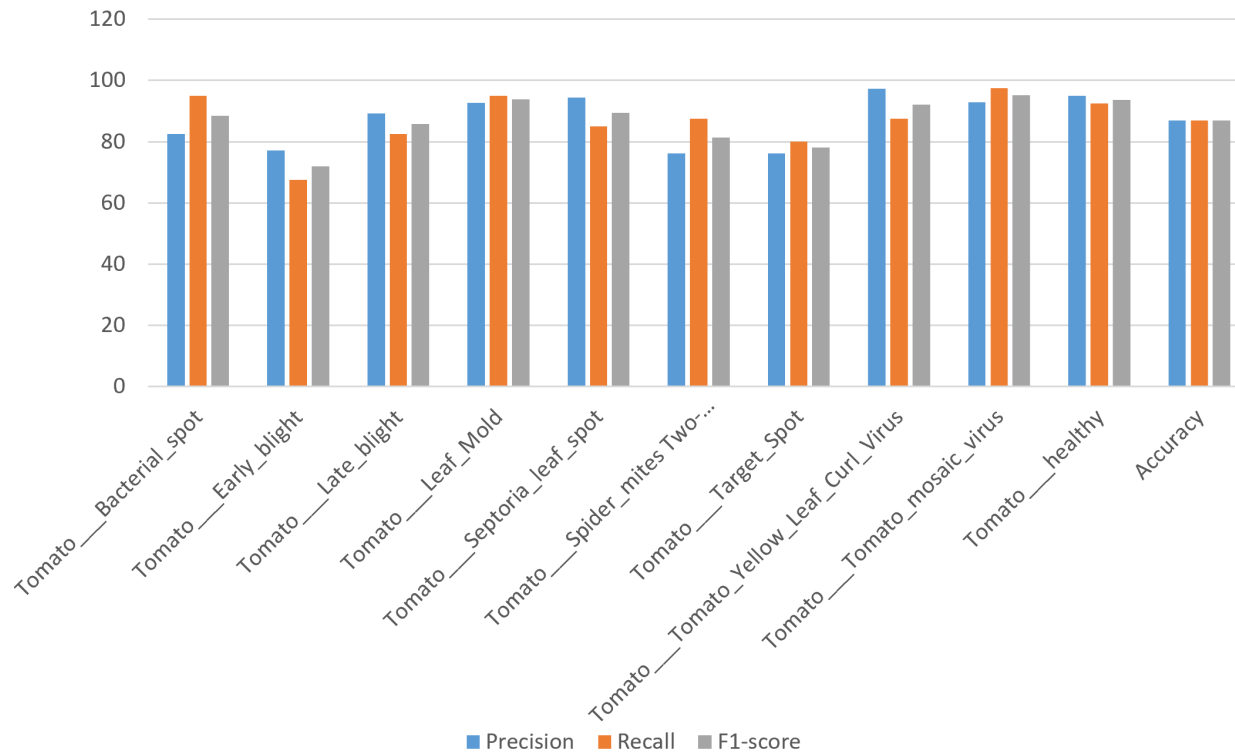


Figure 5: Performance metrics comparison.

Table 5: Repeated-run results (fixed 1,600/400 split, 8 epochs, CHBOA-selected hyperparameters), final-model results, and model parameters.

Setting	Accuracy / parameters
5 runs: mean $\pm$ SD (95% CI)	83.25% $\pm$ 1.22 (81.73–84.77)
10 runs: mean $\pm$ SD (95% CI)	81.95% $\pm$ 1.02 (81.22–82.68)
20 runs: mean $\pm$ SD (95% CI)	81.94% $\pm$ 1.37 (81.29–82.58)
All 35 runs: mean $\pm$ SD (95% CI)	82.13% $\pm$ 1.31 (81.68–82.58)
Total parameters	12,022,347
Trainable parameters	935,270
Non-trainable parameters	11,087,075
Optimizer parameters	2
Final model, 15 epochs (400-image validation split)	87.00%
Final model, independent 1,000-image test set (95% Wilson CI)	87.5% (85.3–89.4)

Table 6: Individual run accuracies (%) of the 35 repeated 8-epoch training runs (same 1,600/400 split, CHBOA-selected hyperparameters).

Group	Run accuracies (%)
5 runs	81.75, 83.50, 82.50, 83.50, 85.00
10 runs	81.50, 81.75, 83.00, 81.75, 80.75, 81.75, 80.75, 82.75, 84.00, 81.50
20 runs	83.25, 83.00, 82.25, 80.00, 80.50, 82.25, 80.75, 82.50, 83.50, 83.50, 81.25, 80.25, 81.50, 81.25, 80.75, 84.75, 79.75, 81.75, 83.00, 83.00

and the learning rate. The per-weight first- and second-moment variables that Adam creates once training starts are not included in this count.

The final model reached 87.0% on the 400-image validation split, compared with 82.1% for the 8-epoch repeated runs. All of these runs used the same CHBOA-selected hyperparameters and the same split. The difference therefore reflects the longer training (15 versus 8 epochs), not the hyperparameter search. Because the same 400 images were used for the search fitness, the repeated runs, and the final evaluation, 87.0% is an optimistic estimate.

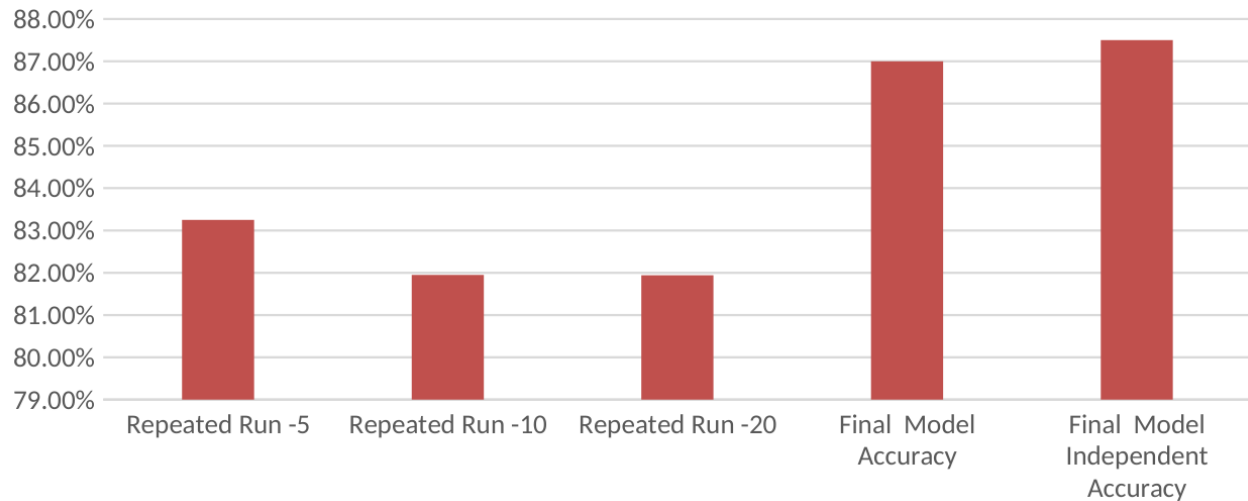


Figure 6: Performance metrics comparison.

Table 7: Per-class results (%) of the final model on the independent 1,000-image test set (100 images per class).

Class	Precision	Recall	F1-score
Tomato___Bacterial_spot	89.9	98	93.8
Tomato___Early_blight	74.5	73	73.7
Tomato___Late_blight	80.5	91	85.4
Tomato___Leaf_Mold	88.4	84	86.2
Tomato___Septoria_leaf_spot	91.6	87	89.2
Tomato___Spider_mites_Two-spotted_spider_mite	80.2	93	86.1
Tomato___Target_Spot	84.3	75	79.4
Tomato___Tomato_Yellow_Leaf_Curl_Virus	100	93	96.4
Tomato___Tomato_mosaic_virus	93.9	92	92.9
Tomato___healthy	94.7	89	91.8
Accuracy	87.5	87.5	87.5
Macro average	87.8	87.5	87.5
Weighted average	87.8	87.5	87.5

3.2. Independent test-set evaluation results

The saved final model was evaluated once on the dataset's separate 1,000-image validation folder (100 images per class), which was not used for training, hyperparameter search, or model selection. On these images it correctly classified 875 of 1,000 images, an accuracy of 87.5% (95% Wilson CI 85.3–89.4%). Its macro-averaged precision, recall, and F1-score were 0.878, 0.875, and 0.875; the weighted averages are identical because every class has 100 images. Per-class results are given in Table 7.

The independent test accuracy (87.5%) is close to the validation accuracy of the same model (87.0%), so the optimistic bias of the validation estimate appears small for this model. The class-level pattern is also consistent: Tomato Yellow Leaf Curl Virus (F1 = 0.964) and Bacterial Spot (F1 = 0.938) are recognised best, and Early Blight (F1 = 0.737) and Target Spot (F1 = 0.794) remain the weakest classes. Nested cross-validation was not performed.

Figure 7 shows the training and validation accuracy (left) and loss (right) of the final model over its 14 training epochs ($\text{lr} = 6.63 \times 10^{-4}$, 404 dense units, dropout 0.571, mini-batch 32, fixed 1,600/400 split; random seeds not set). Training accuracy rose from 0.22 to 0.88 and validation accuracy from 0.63 to 0.87. Training loss fell from 2.29 to 0.38 and validation loss from 1.36 to 0.45. The figure documents this single training run only and is not used to draw conclusions about generalisation.

4. Discussion

Table 8 places the present results alongside accuracies and runtimes reported for ResNet-9 [9], transfer learning [4], and GD-Attention [16]. The published accuracies are 85.6% for ResNet-9, 86.31% for transfer learning, and 87.24% for GD-Attention. The hybrid model achieved a mean of 82.13% over 35 repeated 8-epoch runs, 87.0% for the final 15-epoch model on its validation split, and 87.5% (95% CI 85.3–89.4%) on the independent test set. The literature values were taken from the original studies and were

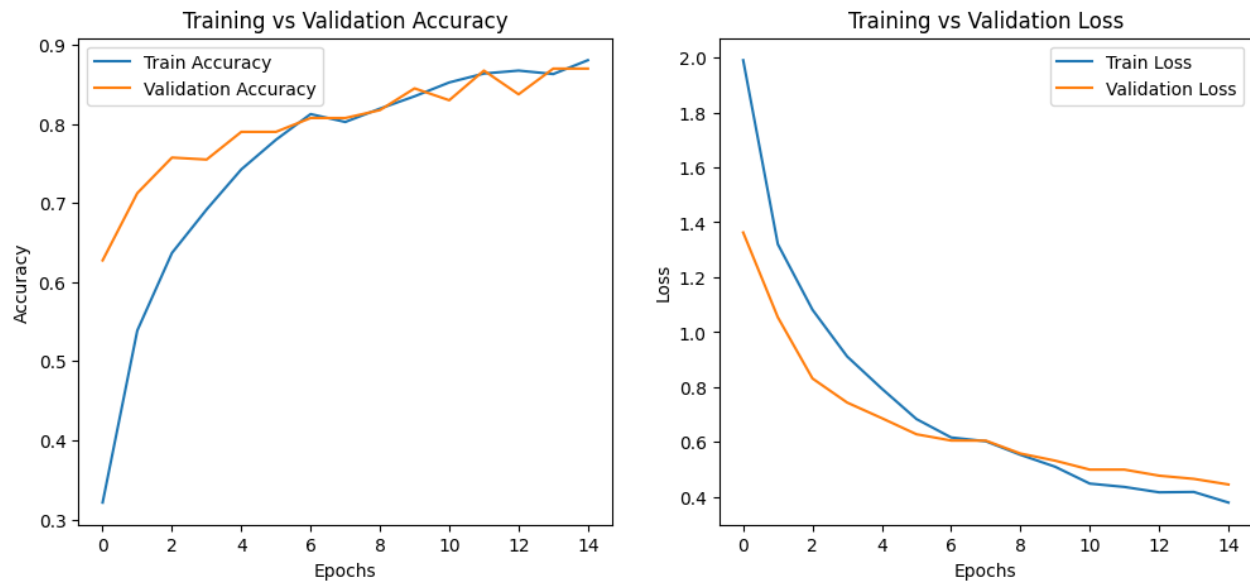


Figure 7: Training versus validation loss and accuracy.

Table 8: Comparative analysis.

Reference	Model	Accuracy (%)	Run time (s)	Evaluation source
[9]	ResNet-9	85.6	187.5	Reported in [9]
[4]	Transfer Learning	86.31	179.2	Reported in [4]
[16]	GD-Attention	87.24	192.6	Reported in [16]
This study (hybrid)	EfficientNetB0 and DenseNet121	82.13	205.3	Mean of 35 repeated 8-epoch runs, 400-image split
This study (CHBOA_Hybrid, final)	CHBOA + EfficientNetB0 and DenseNet121	87.0 (validation split) / 87.5 (independent test)	165.4	15-epoch final model

not re-implemented on the same data subset, splits, preprocessing, hardware, epoch count, or timing protocol. They are therefore non-controlled contextual figures, and Table 8 does not establish that any method is more accurate or faster than another. The 165.4s reported for the final model covers final-model training only and excludes the 25 CHBOA fitness evaluations, whose wall-clock time was not recorded.

The study also has five limitations as follows:

1. No cross-validation was performed: the repeated runs reuse one fixed training/validation split, so they measure variability between training runs, not variability across data partitions.
2. The validation figures are optimistic: the same 400 validation images were used for the CHBOA fitness, for the repeated runs, and for the final evaluation, and only the independent test result (87.5%) is free of selection bias.
3. Batch size was not actually searched: the batch-size variable changed only the number of training steps per fitness evaluation, while the mini-batch size stayed at 32. Candidates with $bs > 50$ were trained on the full training partition, whereas the others were trained on only 1–3 batches per epoch. The selected configuration ($bs = 58$) was among those trained on the full partition, so the search compared unequally trained candidates; its best fitness was also found in the first iteration and never improved, and the benefit of the search over default hyperparameters is therefore not established.
4. The random draws of the search were not seeded, so the search cannot be repeated exactly.
5. Only 2,000 of the 11,000 available laboratory-style images were used for training and tuning, no augmentation was applied, and performance under field conditions is untested.

5. Conclusion

Using DenseNet121 and EfficientNetB0 as frozen feature extractors, this deep learning hybrid model classifies tomato leaf diseases. Over 35 repeated 8-epoch training runs on a fixed split, the model achieved a mean validation accuracy of 82.1% (SD 1.3

percentage points). By utilising the backbones' pre-trained weights, only approximately 935K of more than 12 million parameters need to be trained, which improves training efficiency.

With the CHBOA-selected learning rate, dense units, and dropout rate, a final model trained for 15 epochs reached 87.0% on the validation split and 87.5% (95% CI 85.3–89.4%) on an independent 1,000-image test set. For the majority of diseases, including viruses (e.g., Mosaic Virus and Tomato Yellow Leaf Curl Virus) and bacterial spot, the per-class evaluation showed high recall, precision, and F1-scores. Early blight and target spot remained difficult to separate because of their visual similarity. Future work will use genuine stratified cross-validation with fold-level logging, a seeded search in which every candidate receives the same training budget and batch size is actually varied, data augmentation, the full dataset, and field images.

Data availability

The data supporting the findings of this study are available from the corresponding author on request.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this manuscript.

Funding

The authors received no specific funding from any public, commercial, or not-for-profit organisation for this research.

Acknowledgment

The authors thank the Department of Computer Science at the African University of Science and Technology, Abuja, Nigeria.

References

- [1] W. Albattah, A. Javed, M. Nawaz, M. Masood & S. Albahli, "Artificial intelligence-based drone system for multiclass plant disease detection using an improved efficient convolutional neural network", *Frontiers in Plant Science* **13** (2022) 808380. <https://doi.org/10.3389/fpls.2022.808380>.
- [2] G. Sekar & P. M. Benson Mansingh, "Smart leaf recognition system", *International Journal of Engineering and Advanced Technology* **9** (2020) 289. <https://doi.org/10.35940/ijeat.d7577.069520>.
- [3] U. Barman, P. Sarma, M. Rahman, V. Deka, S. Lahkar, V. Sharma & M. J. Saikia, "ViT-SmartAgri: vision transformer and smartphone-based plant disease detection for smart agriculture", *Agronomy* **14** (2024) 327. <https://doi.org/10.3390/agronomy14020327>.
- [4] M. S. A. M. Al-gaashani, F. Shang, M. S. A. Muthanna, M. Khayyat & A. A. Abd El-Latif, "Tomato leaf disease classification by exploiting transfer learning and feature concatenation", *IET Image Processing* **16** (2022) 913. <https://doi.org/10.1049/ipr2.12397>.
- [5] M. S. Islam, S. Sultana, F. A. Farid, M. N. Islam, M. Rashid, B. S. Bari, N. Hashim & M. N. Husen, "Multimodal hybrid deep learning approach to detect tomato leaf disease using attention-based dilated convolution feature extractor with logistic regression classification", *Sensors* **22** (2022) 6079. <https://doi.org/10.3390/s22166079>.
- [6] S. K. Hosseiny, N. Jola & S. M. Hosseiny, "Diagnosis of leaf surface disease using two datasets of tomato and rice obtained from image processing techniques", *Review of Computer Engineering Studies* **8** (2021) 77. <https://doi.org/10.18280/rces.080303>.
- [7] H. Sabrol & K. Satish, "Tomato plant disease classification in digital images using classification tree", *International Conference on Communication and Signal Processing (ICCSP)*, Melmaruvathur, India, 2016, pp. 1242–1246. <https://doi.org/10.1109/ICCSP.2016.7754351>.
- [8] R. Kumar, A. Kumar, K. Bhatia, K. S. Nisar, S. S. Chouhan, P. Maratha & A. K. Tiwari, "Hybrid approach of cotton disease detection for enhanced crop health and yield", *IEEE Access* **12** (2024) 132495. <https://doi.org/10.1109/ACCESS.2024.3419906>.
- [9] A. O. Anim-Ayeko, C. Schillaci & A. Lipani, "Automatic blight disease detection in potato (*Solanum tuberosum* L.) and tomato (*Solanum lycopersicum* L. 1753) plants using deep learning", *Smart Agricultural Technology* **4** (2023) 100178. <https://doi.org/10.1016/j.atech.2023.100178>.
- [10] S. S. Chouhan, U. P. Singh, U. Sharma & S. Jain, "Classification of different plant species using deep learning and machine learning algorithms", *Wireless Personal Communications* **136** (2024) 2275. <https://doi.org/10.1007/s11277-024-11374-y>.
- [11] S. S. Chouhan, U. P. Singh, U. Sharma & S. Jain, "Leaf disease segmentation and classification of *Jatropha curcas* L. and *Pongamia pinnata* L. biofuel plants using computer vision-based approaches", *Measurement* **171** (2021) 108796. <https://doi.org/10.1016/j.measurement.2020.108796>.
- [12] S. Albahli & M. Nawaz, "DCNet: DenseNet-77-based CornerNet model for the tomato plant leaf disease detection and classification", *Frontiers in Plant Science* **13** (2022) 957961. <https://doi.org/10.3389/fpls.2022.957961>.
- [13] L. Tan, J. Lu & H. Jiang, "Tomato leaf diseases classification based on leaf images: a comparison between classical machine learning and deep learning methods", *AgriEngineering* **3** (2021) 542. <https://doi.org/10.3390/agriengineering3030035>.
- [14] S. A. Wagle & R. Harikrishnan, "A deep learning-based approach in classification and validation of tomato leaf disease", *Traitement du Signal* **38** (2021) 699. <https://doi.org/10.18280/ts.380317>.
- [15] O. B. Ayode, M. O. Raji, A. A. Akindele, K. J. Yusuf-Mashopa, M. F. Abdulrauf, I. A. Raji & F. B. Musah, "Hyperparameter optimisation for support vector machine-based disease detection in maize leaf variants", *African Scientific Reports* **4** (2025) 320. <https://doi.org/10.46481/asr.2025.4.3.320>.
- [16] Z. Li, W. Tao, J. Liu, F. Zhu, G. Du & G. Ji, "Tomato leaf disease recognition via optimizing deep learning methods considering global pixel value distribution", *Horticulturae* **9** (2023) 1034. <https://doi.org/10.3390/horticulturae9091034>.
- [17] A. Das, F. Pathan, J. R. Jim, M. M. Kabir & M. F. Mridha, "Deep learning-based classification, detection, and segmentation of tomato leaf diseases: a state-of-the-art review", *Artificial Intelligence in Agriculture* **15** (2025) 192. <https://doi.org/10.1016/j.aiaa.2025.02.006>.

- [18] A. Bajpai & T. Tyagi, “An efficient approach to detect and predict the tomato leaf disease using enhance segmentation neural network”, *SN Computer Science* **4** (2023) 795. <https://doi.org/10.1007/s42979-023-02262-6>.
- [19] R. Khan, N. Ud Din, A. Zaman & B. Huang, “Automated tomato leaf disease detection using image processing: an SVM-based approach with GLCM and SIFT features”, *Journal of Engineering* **2024** (2024) 9918296. <https://doi.org/10.1155/2024/9918296>.
- [20] B. Kaustubh, “Tomato leaf disease detection”, Kaggle (2020). Available online: <https://www.kaggle.com/datasets/kaustubhb999/tomatoleaf>.
- [21] S. Khan & M. Narvekar, “Novel fusion of color balancing and superpixel based approach for detection of tomato plant diseases in natural complex environment”, *Journal of King Saud University – Computer and Information Sciences* **34** (2022) 3506. <https://doi.org/10.1016/j.jksuci.2020.09.006>.
- [22] M. Tan & Q. V. Le, “EfficientNet: rethinking model scaling for convolutional neural networks”, *Proceedings of the 36th International Conference on Machine Learning (ICML)*, Long Beach, USA, (2019), pp. 6105–6114. <https://doi.org/10.48550/arXiv.1905.11946>.
- [23] G. Huang, Z. Liu, L. Van Der Maaten & K. Q. Weinberger, “Densely connected convolutional networks”, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, USA, (2017), pp. 2261–2269. <https://doi.org/10.1109/CVPR.2017.243>.
- [24] F. A. Hashim, E. H. Houssein, K. Hussain, M. S. Mabrouk & W. Al-Atabany, “Honey badger algorithm: new metaheuristic algorithm for solving optimization problems”, *Mathematics and Computers in Simulation* **192** (2022) 84. <https://doi.org/10.1016/j.matcom.2021.08.013>.
- [25] H. Givi & M. Hubalovska, “Skill optimization algorithm: a new human-based metaheuristic technique”, *Computers, Materials & Continua* **74** (2023) 179. <https://doi.org/10.32604/cmc.2023.030379>.