computers

MDPI

Article

A Weighted Ensemble of Convolutional Neural Networks for Anthracnose Detection in Avocado Fruit

Anibal Flores [1,*], Jose Guzman-Valdivia [1], Saul Huaquipaco [1], Hugo Tito-Chura [1], Ruso Morales-Gonzales [1], Carlos Silva-Delgado [1] and Eduardo Flores-Quispe [2]

[1] Departamento Académico de Ingeniería de Sistemas e Informática, Universidad Nacional de Moquegua, Moquegua 18611, Peru; jguzmanv@unam.edu.pe (J.G.-V.); shuaquipacoe@unam.edu.pe (S.H.); etitoc@unam.edu.pe (H.T.-C.); rmoralesg@unam.edu.pe (R.M.-G.); csilvad@unam.edu.pe (C.S.-D.)
[2] Departamento Académico de Ingeniería Ambiental, Universidad Nacional de Moquegua, Moquegua 18611, Peru; efloresq@unam.edu.pe
* Correspondence: afloresg@unam.edu.pe

Abstract

Global avocado production exceeds 10 million tons annually. Among the diseases affecting avocado fruit, anthracnose is one of the most significant, causing black lesions and fruit decay that can result in yield losses of 20–30%. To facilitate the early detection of anthracnose, this study proposes a computer vision-based approach. A dataset containing 2218 images of Fuerte avocados was first developed, comprising 1730 healthy samples and 488 anthracnose-infected samples after the labeling process. In the experimental phase, several convolutional neural network (CNN) models with varying depths (3, 4, 5, and 6 layers) were designed and evaluated. These models were subsequently integrated into different weighted ensemble configurations, where the best performance was achieved by the ensemble combining all four individual CNNs. The proposed weighted ensemble was compared against widely used state-of-the-art architectures, including VGG-16, ResNet-18, and MobileNetV2. Experimental results demonstrated the effectiveness of the proposed approach, achieving an F1-score of 0.9052, outperforming VGG-16 (0.8283), ResNet-18 (0.7328), and MobileNetV2 (0.7320).

Keywords: weighted ensemble model; convolutional neural networks; anthracnose detection; avocado disease

Academic Editors: Paolo Bellavista, Lei Zhou and Zhenchang Xia

Received: 25 April 2026
Revised: 6 June 2026
Accepted: 7 June 2026
Published: 10 June 2026

1. Introduction

Avocado (*Persea americana*) is one of the most economically important fruit crops in Latin America, particularly in countries such as Mexico, Peru, and Chile [1]. Its economic relevance stems from its high commercial value and its role as a major agricultural export product.

Avocado production is affected by a wide range of diseases [2] caused by fungi, bacteria, viruses, and insects. Among the most important pathogens are *Phytophthora cinnamomi*, *Colletotrichum gloeosporioides*, *Sphaceloma perseae*, and *Oidium* spp. This study focuses on anthracnose, one of the most prevalent and destructive fungal diseases affecting avocado fruits. Anthracnose is primarily associated with fungi belonging to the genera Colletotrichum and Gloeosporium and can significantly reduce fruit quality and marketability.

Early detection of anthracnose [3] is particularly important because symptoms often remain latent during cultivation and become visible only during postharvest storage and transportation [4]. Consequently, infected fruits may be distributed together with healthy

ones, causing economic losses throughout the supply chain. Automated detection systems can assist producers and distributors by identifying infected fruits before commercialization and preventing the spread of damage within storage and transport batches [5].

To address this challenge, this study proposes a weighted ensemble model based on convolutional neural networks (CNNs) for anthracnose detection in avocado fruits. The proposed approach is evaluated against widely used benchmark architectures, including VGG-16 [6], ResNet-18 [7], and MobileNetV2 [8]. For this purpose, a dedicated dataset containing 2218 images of healthy and anthracnose-infected Fuerte avocados was developed.

The methodological contributions of this work are based on the following aspects:

- Development of a dedicated dataset comprising 2218 images of healthy and anthracnose-infected Fuerte avocado fruits collected under real conditions.
- Design of a computationally efficient disease-detection framework based on shallow convolutional neural networks, providing an alternative to deeper and more resource-intensive architectures.
- Implementation of a class-imbalance handling strategy to improve the learning process and classification performance.
- Development and experimental optimization of a weighted ensemble model that combines multiple shallow CNN architectures for anthracnose detection.
- Comprehensive comparative evaluation against state-of-the-art CNN architectures, including VGG-16, ResNet-18, and MobileNetV2, supported by statistical significance analysis.

The literature review identified eight studies addressing disease detection in avocado leaves and fruits. Among them, only one study focused specifically on anthracnose detection and employed a dataset of 674 images collected from publicly available online repositories. In contrast, the present study introduces a newly developed dataset comprising 2218 images of Fuerte avocados, providing a substantially larger and more specialized resource for anthracnose detection research.

2. Related Works

For the literature review, studies indexed in Scopus between 2015 and 2025 were considered. The inclusion criteria comprised publications focused on the detection, classification, or diagnosis of diseases affecting fruit crops or fruit trees using computer vision, machine learning, deep learning, RGB imaging, multispectral imaging, or related technologies. Studies exclusively devoted to pathogen identification, agronomic management practices, epidemiological analyses, or approaches lacking automated detection methods were excluded.

A total of 33 relevant studies were identified. These studies employed a wide range of techniques, ranging from traditional machine learning algorithms to advanced deep learning architectures, for disease detection in fruits and other plant organs across different crop species. However, only eight studies specifically addressed avocado-related diseases. Among these, five focused on laurel wilt disease affecting different parts of the avocado plant, whereas only three investigated diseases occurring directly on avocado fruits. This finding reveals the limited attention that fruit disease detection in avocado has received within the existing literature.

Among the studies focused on avocado fruits, [9] investigated the detection of internal rot using X-ray imagery and reported an accuracy of 98% with the U-Net++ architecture. Study [10] addressed stem-end rot detection in Hass avocados using a dataset of 560 images and machine learning models such as Random Forest and XGBoost, achieving accuracies ranging from 56% to 65%. More recently, [11] employed a dataset of 674 images collected from publicly available repositories, including Kaggle, and evaluated several deep learning

architectures, such as ResNet50, InceptionV3, EfficientNetB2, VGG16, and DenseNet121. Their best-performing model achieved an accuracy of 99.03%, a precision of 98.92%, and a recall of 98.89%.

Table 1 summarizes the related works on the topic of this manuscript.

Table 1. Summary of Related works.

Work	Crop	Disease	Part of the Plant	Data Acquisition	Technique	Result
[12]	Avocado	Laurel wilt	Canopy, orchard	Multispectral	MLP	In total, 91 to 100% overall classification
[13]	Avocado	Laurel wilt	trees	Aerial images	ANOVA, M-statistic	
[14]	Apple	Marssonina blotch	Leaves	Hyperspectral images	Ensemble bagged, decision tree, weighted KNN	
[15]	Mango	Any disease	Leaves	Mendeley dataset	Resnet-50, SVM, Gradient boosting, logistic regression, XGBoost, Decision Tree, KNN	
[16]	Avocado	Laurel wilt, nutritional deficiency	Leaves	Hyperspectral images	FDA-BC	Acc = 100%
[17]	Avocado	Laurel wilt, phytophthora root rot, nutritional stresses	Top view of the trees	RGB images	ANN	Acc = 99.4%
[18]	Apple	Any affecting	leaves	Termal camera	CNN	
[19]	Apple	Fire blight	Leaves, tree canopy	New plant diseases dataset	SVM, Mask R-CNN	
[20]	Apple and pears	Pear black spot, pear rust, apple mosaic, apple rust	leaves	AKSS dataset	Ensemble learning, ResNet-101, DenseNet-121	
[21]	Walnut	Anthracnose infection	Tree canopy	RGB images, VOC2027 dataset	SSD	
[22]	Citrus	Gummosis	Orchard	Multispectral and hyperspectral images	Supervised Spectral Angle Mapper	
[23]	Apple	Black spot, black rot, and cedar rust	Leaves	RGB images	SVM	
[24]	Apple	Apple scab, black rot, cedar rust	Leaves	New plant diseases dataset	CNN	
[25]	Sweet Cherry	Any kind of stress or disease	Leaves and branches	ERICA dataset	YoloV5m	
[26]	Apple	Moldy core	Fruits	Hyperspectral spectroscopy images	ANN, bagging	
[27]	Apple	Alternaria leaf blotch	Leaves	RGB images	DeeplabV3+, PSPNet and UNet	
[28]	Apple	Valsa canker	Tree branches	Hyperspectral images	Dual Channel CNN	
[28]	Apple	Valsa canker	Branches and trunk	Hyperspectral images	Mahalanobis distance	
[29]	Apple	Fire blight	Orchard	Multi-spectral images	Decision trees, random forest, SVM	
[30]	Mango	Anthracnose	Leaves	Plantvillage dataset	CNN	
[31]	Citrus	Huanglongbing	Leaves	RGB images	AlexNet, VGG16, VGG19, ResNet18, GoogleNet, Inception-V3	
[32]	Pears	Moldy core	Fruits	Spectroscopy images	SVM, LS SVM, Random Forest,	

Table 1. *Cont.*

Work	Crop	Disease	Part of the Plant	Data Acquisition	Technique	Result
[33]	Plantain tree	Xanthomonas wilt, bunchy top, black sigatoka, yellow sigatoka	Whole tree	RGB images	RNN + CNN	
[34]	Banana	Banana fusarium wilt	Orchard	Multispectral images	SVM, Random Forests, BPNN, Logistic regression	
[35]	Stone fruit	Bacterial and fungal	Leaves	RGB images	KNN, SGD, Random Forests	
[36]	Apple	Apple scab	Leaves	RGB images	Mask R-CNN	
[37]	Apple	Foliar disease	Leaves	FGVC8 dataset	CNN, logistic regression, random forest, decision tree	
[38]	Citrus	Huanglongbing	Fruits	RGB images	YOLO and Faster R-CNN, MobileNetV2, EfficientNetV2-BO, NASNet-Mobile	
[39]	Oil palm	Basal stem rot	Canopy	RGB images	M-CR U-Net	
[9]	Avocado	Internal rot	Fruit	X-ray images	U-Net++	Acc = 98.00%
[40]	Avocado, citrus	Laurel wilt, citrus canker	Leaves	Hyperspectral images	KLE	Acc = 99.00% to 100.00%
[10]	Avocado	Stem-end rot	Fruit		Random Forest, XGBoost	Acc = 56.00% to 65.00%
[11]	Avocado	Anthracnose and scab	Fruit	Kaggle and others	VotingBS	Acc = 99.03% Prec = 98.92% Rec = 98.89%

Among the 33 selected studies, accuracy was the most frequently reported evaluation metric. However, only a limited number of studies provided complementary performance indicators such as precision, recall, F1-score, sensitivity, specificity, or statistical significance analyses. This observation suggests that model reliability and robustness are often insufficiently assessed in the existing literature, particularly in scenarios involving class imbalance, where accuracy alone may provide an incomplete representation of model performance.

Furthermore, many studies primarily focused on demonstrating proof-of-concept results under controlled experimental conditions. Consequently, the generalization capability and practical applicability of the proposed methods in real-world agricultural environments remain largely unexplored. These limitations highlight the need for more comprehensive evaluation frameworks that consider multiple performance metrics, dataset characteristics, and operational constraints.

Based on the literature review, several research gaps were identified regarding avocado fruit disease detection. In particular, the limited availability of dedicated datasets, the scarcity of studies focused on anthracnose detection, and the insufficient assessment of model reliability motivated the development of the proposed approach. Table 2 summarizes the main differences between the related studies and the present work, highlighting the identified research gaps and the contributions of this study.

Table 2. Differences between related works for avocado fruit vs. proposal.

Related Works	This Study
Most studies focus on leaf diseases such as laurel wilt.	This study focuses on anthracnose in avocado fruits.
The related work on anthracnose detection in avocado fruits uses complex models such as ResNet50, VGG16, InceptionV3, EfficientNetB2, and DenseNet121.	The study proposes a weighted ensemble model based on traditional CNNs.
The exact country of origin of the images is unknown, as they were obtained from different repositories.	It is the first study conducted using data obtained in Peru.
The related work focuses on the Hass variety and others.	The study focuses exclusively on the Fuerte variety.

3. Methodology

Figure 1 summarizes the process for the implementation of the proposed model and the various models under study.

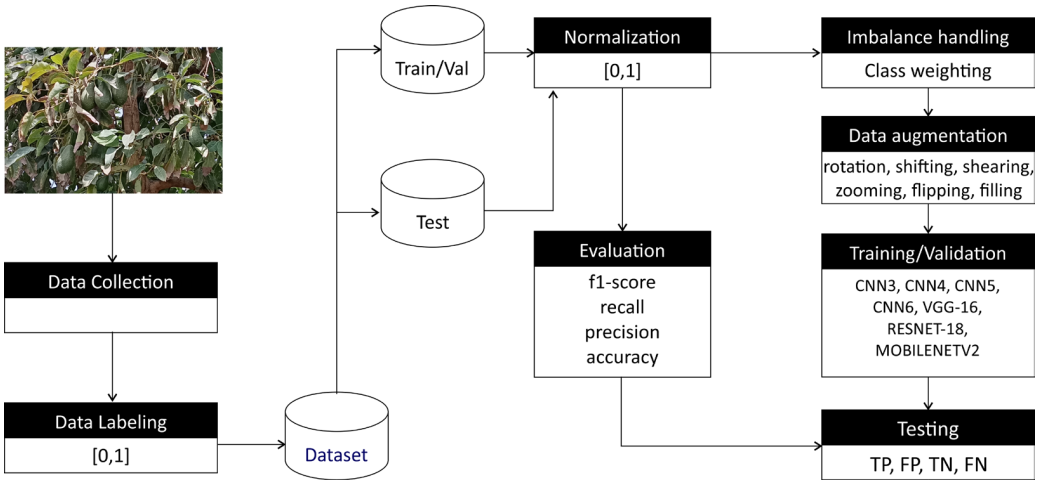

Figure 1. Implementation process of the studied models.

The process of this study is summarized in Figure 1. Initially, the acquisition of Fuerte avocado images is carried out; these images are then labeled, followed by the creation of the training, validation, and test partitions. Subsequently, the architectures and various models are implemented and evaluated using F1-score, recall, precision, and accuracy.

3.1. Data Collection

The constructed dataset consists of 2218 images. Images of fruits collected from different avocado trees located in the Moquegua Valley, Mariscal Nieto Province, Moquegua Region, Peru, were used. These images were collected between April and July 2025. The dataset includes images captured with a 50 MP camera with an f/18 aperture, at a resolution of 3072 × 4080 pixels, against a gray background in JPG format, in an environment with natural lighting conditions. For the implementation of the proposed model and the baseline models, the images were resized to 224 × 224 pixels in the same format.

3.2. Labeling

The labeling was carried out by a specialist in avocado diseases. Following the labeling process, of the 2218 images, 1730 correspond to the healthy class or 0, meaning the fruit shows no anthracnose disease, and 488 correspond to the unhealthy class or 1, meaning the fruit presents anthracnose.

Some examples of the labeling results are shown in Figure 2.

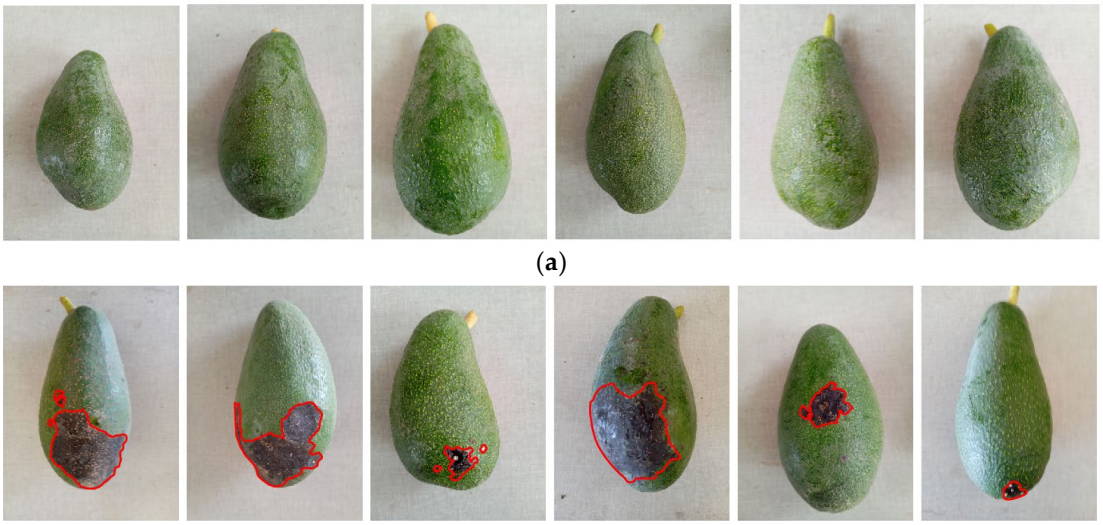

Figure 2. Sample images from both classes. (**a**) Healthy, and (**b**) Unhealthy, the affected area is outlined in red.

3.3. Data Subsets

For the experiments, the data were divided into training, validation, and test sets. Of these, 1572 images correspond to the training set, 216 to the validation set, and 430 to the test set. Of the 1572 training images, 1242 belong to the healthy class and 330 to the unhealthy class. Of the 216 validation images, 170 belong to the healthy class and 46 to the unhealthy class. Finally, of the 430 test images, 318 belong to the healthy class and 112 to the unhealthy class. Table 3 shows a summary of elements per class in each subset.

Table 3. Summary of elements per class in each subset.

Class	Training	Validation	Test
Healthy (0)	1242	170	318
Unhealthy (1)	330	46	119

3.4. Handling Class Imbalance

Various strategies can address class imbalance, such as increasing the minority class (over-sampling), reducing the majority class (under-sampling), and class weighting. This study employs class weighting, a mathematical technique used during machine learning and deep learning model training. Class weighting modifies the loss function so that instead of all errors having the same cost, a different penalty or weight is assigned based on the class. Consequently, the majority class receives a low weight, while the minority class receives a high weight.

The weight of each class w_c is estimated using Equation (1).

$$w_c = \frac{n_s}{n_c \times nsc} \tag{1}$$

where

n_s is the number of samples of the training subset;
n_c is the number of classes;
nsc is the number of samples of the class.

According to Table 3 and Equation (1), the class weights for classes 0 and 1 within the training subset are obtained, as presented in Table 4.

Table 4. Estimated weights for each class in the training subset.

Class	Class Weight Estimation	Class Weight
Healthy (0)	$w_0 = \frac{1572}{2 \times 1242}$	1.2637
Unhealthy (1)	$w_1 = \frac{1572}{2 \times 330}$	4.7349

3.5. Data Augmentation

Data augmentation enables the generation of artificial variations in training images in order to increase data diversity and improve the model's generalization capability. In this case, the training set contains 1572 original images, which were dynamically transformed during training through different geometric and spatial operations. The Python 3.11.7 code block shown in Figure 3 enables data augmentation on the training data.

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

Figure 3. Python code for data augmentation.

rescale=1./255: normalizes pixel values to the range [0,1] by dividing each value by 255. rotation_range=20: randomly rotates each image by up to ±20 degrees. width_shift_range=0.2: horizontally shifts the image by up to 20% of its width. height_shift_range=0.2: vertically shifts the image by up to 20% of its height. shear_range=0.2: applies shear transformations to the image. zoom_range=0.2: performs random zoom-in or zoom-out transformations of up to 20%. horizontal_flip=True: horizontally flips some images. fill_mode='nearest': fills empty pixels generated by the transformations using the value of the nearest pixel.

This generator does not physically create new images stored on disk. Instead, during each epoch, it produces new transformed versions of the 1572 original images in real time (on-the-fly). As a result, the model observes different variations in the same image across different training iterations, which reduces overfitting and improves robustness to changes in orientation, position, and scale.

In this study, the training process is carried out over 50 epochs; therefore, the model will not only see 1572 fixed images, but also thousands of dynamically generated variants derived from them. This effectively increases the diversity of the training set without the need to collect new real images.

3.6. Modeling

The basic CNN models were implemented in Jupyter 7.2.2 IDE using the TensorFlow 2.18.0 library in Python 3.11.7; their respective architectures are shown in Figure 4.

Layer	Output Shape	Param #
conv2d_1(Conv2D)	(None, 224,224,32)	896
batch_normalization_1 (BatchNormalization)	(None, 224,224,32)	128
max_pooling2d_1(MaxPooling2D)	(None, 112,112,32)	0
conv2d_2(Conv2D)	(None, 112,112,64)	18,496
batch_normalization_2 (BatchNormalization)	(None, 112,112,64)	256
max_pooling2d_2 (MaxPooling2D)	(None, 56,56,64)	0
conv2d_3(Conv2D)	(None, 56,56,128)	73,856
batch_normalization_3 (BatchNormalization)	(None, 56,56,128)	512
max_pooling2d_3 (MaxPooling2D)	(None, 28,28,128)	0
global_average_pooling2d_1 (GlobalAveragePooling2d)	(None, 128)	0
dense_1(Dense)	(None, 128)	16,512
dropout_1(Dropout)	(None, 128)	0
dense_2(Dense)	(None, 2)	258

(**a**)

Layer	Output Shape	Param #
conv2d_1(Conv2D)	(None, 224,224,32)	896
batch_normalization_1 (BatchNormalization)	(None, 224,224,32)	128
max_pooling2d_1(MaxPooling2D)	(None, 112,112,32)	0
conv2d_2(Conv2D)	(None, 112,112,64)	18,496
batch_normalization_2 (BatchNormalization)	(None, 112,112,64)	256
max_pooling2d_2 (MaxPooling2D)	(None, 56,56,64)	0
conv2d_3(Conv2D)	(None, 56,56,128)	73,856
batch_normalization_3 (BatchNormalization)	(None, 56,56,128)	512
max_pooling2d_3 (MaxPooling2D)	(None, 28,28,128)	0
conv2d_4(Conv2D)	(None, 28,28,256)	295,168
batch_normalization_4 (BatchNormalization)	(None, 28,28,256)	1024
max_pooling2d_4 (MaxPooling2D)	(None, 14,14,256)	0
global_average_pooling2d_1 (GlobalAveragePooling2d)	(None, 256)	0
dense_1(Dense)	(None, 128)	32,896
dropout_1(Dropout)	(None, 128)	0
dense_2(Dense)	(None, 2)	258

(**b**)

Layer	Output Shape	Param #
conv2d_1(Conv2D)	(None, 224,224,32)	896
batch_normalization_1 (BatchNormalization)	(None, 224,224,32)	128
max_pooling2d_1(MaxPooling2D)	(None, 112,112,32)	0
conv2d_2(Conv2D)	(None, 112,112,64)	18,496
batch_normalization_2 (BatchNormalization)	(None, 112,112,64)	256
max_pooling2d_2 (MaxPooling2D)	(None, 56,56,64)	0
conv2d_3(Conv2D)	(None, 56,56,128)	73,856
batch_normalization_3 (BatchNormalization)	(None, 56,56,128)	512
max_pooling2d_3 (MaxPooling2D)	(None, 28,28,128)	0
conv2d_4(Conv2D)	(None, 28,28,256)	295,168
batch_normalization_4 (BatchNormalization)	(None, 28,28,256)	1024
max_pooling2d_4 (MaxPooling2D)	(None, 14,14,256)	0
conv2d_5(Conv2D)	(None, 14,14,128)	295,040
batch_normalization_5 (BatchNormalization)	(None,14,14,128)	512
max_pooling2d_5 (MaxPooling2D)	(None, 7,7,128)	0
global_average_pooling2d_1 (GlobalAveragePooling2d)	(None, 128)	0
dense_1(Dense)	(None, 128)	16,512
dropout_1(Dropout)	(None, 128)	0
dense_2(Dense)	(None, 2)	258

(**c**)

Layer	Output Shape	Param #
conv2d_1(Conv2D)	(None, 224,224,32)	896
batch_normalization_1 (BatchNormalization)	(None, 224,224,32)	128
max_pooling2d_1(MaxPooling2D)	(None, 112,112,32)	0
conv2d_2(Conv2D)	(None, 112,112,64)	18,496
batch_normalization_2 (BatchNormalization)	(None, 112,112,64)	256
max_pooling2d_2 (MaxPooling2D)	(None, 56,56,64)	0
conv2d_3(Conv2D)	(None, 56,56,128)	73,856
batch_normalization_3 (BatchNormalization)	(None, 56,56,128)	512
max_pooling2d_3 (MaxPooling2D)	(None, 28,28,128)	0
conv2d_4(Conv2D)	(None, 28,28,256)	295,168
batch_normalization_4 (BatchNormalization)	(None, 28,28,256)	1024
max_pooling2d_4 (MaxPooling2D)	(None, 14,14,256)	0
conv2d_5(Conv2D)	(None, 14,14,128)	295,040
batch_normalization_5 (BatchNormalization)	(None,14,14,128)	512
max_pooling2d_5 (MaxPooling2D)	(None, 7,7,128)	0
conv2d_6(Conv2D)	(None,7,7,128)	147,584
batch_normalization_6 (BatchNormalization)	(None, 7,7,128)	512
max_pooling2d_6 (MaxPooling2D)	(None, 3,3,128)	0
global_average_pooling2d_1 (GlobalAveragePooling2d)	(None, 128)	0
dense_1(Dense)	(None, 128)	16,512
dropout_1(Dropout)	(None, 128)	0
dense_2(Dense)	(None, 2)	258

(**d**)

Figure 4. The architecture of the basic CNN models. (**a**) CNN3, (**b**) CNN4, (**c**) CNN5, and (**d**) CNN6. Param # represents number of parameters.

The architectures of the basic CNN models consist of different convolutional blocks—3, 4, 5, and 6—depending on the model. Each model takes as input images of 224 × 224 pixels with 3 channels (RGB).

Each convolutional block contains a Conv2D layer that applies a number of filters to detect simple features such as edges and basic textures. It also includes a BatchNormalization layer that normalizes the activations to stabilize and accelerate training, and a MaxPooling2D layer that reduces the feature map by half (112 × 112).

The GlobalAveragePooling2D block, instead of flattening the tensor, computes the spatial average of each feature map, producing a vector of 256 values. This drastically reduces the number of parameters and helps prevent overfitting.

The final block corresponds to the classifier, which contains a fully connected Dense layer that learns combinations of the extracted features. The Dropout layer randomly deactivates a percentage of neurons during training to regularize the model and reduce overfitting. The output Dense layer produces a probability per class; in this study, the number of classes is 2.

All models were trained for 50 epochs, using Adam as the optimizer, a learning rate of 0.0001, and early stopping.

In summary, Table 5 presents the hyperparameters of the implemented CNN models.

Table 5. Summary of hyperparameters for the implemented CNN models.

Hyperparameter	Value
Optimizer	Adam
Learning rate	0.0001
Batch size	32
Epochs	50
Input size	224 × 224
Dropout	0.5
Loss function	Categorical Cross-Entropy
Activation	ReLU/Softmax

3.7. Evaluation

The results of the implemented models were evaluated using *Accuracy*, *Precision*, *Recall*, and *F1-Score*, which are estimated through Equations (2)–(5).

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)} \tag{2}$$

$$Precision = \frac{TP}{(TP + FP)} \tag{3}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{4}$$

$$F1 - Score = \frac{2 * Precision * Recall}{(Precision + Recall)} \tag{5}$$

where *TP* are the True Positives, *FP* the False Positives, *TN* the True Negatives, and *FN* the False Negatives.

4. Results and Discussions

4.1. Results

The confusion matrices of the four models based on convolutional neural networks are shown in Figure 5. Likewise, the respective ROC curves of these models are shown in Figure 6.

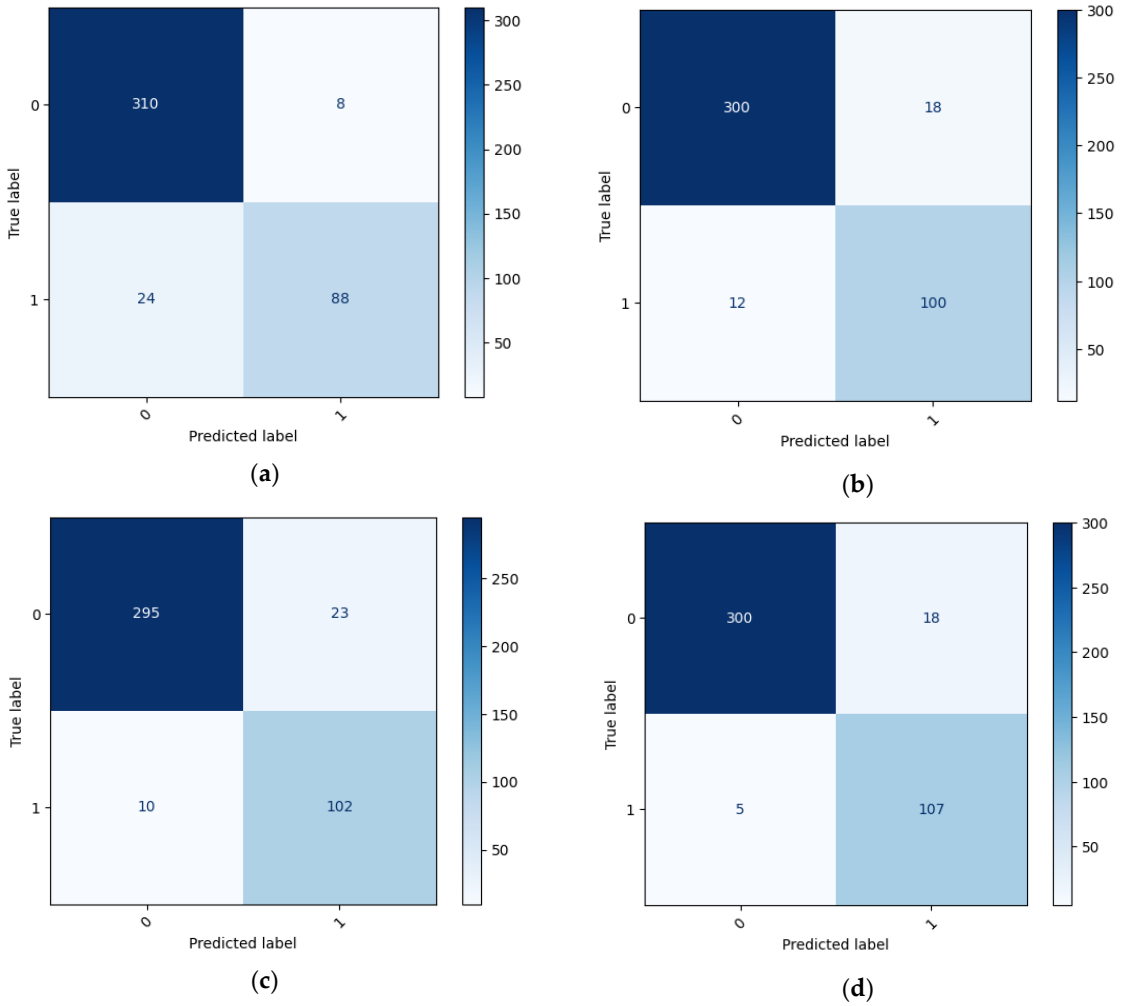

Figure 5. Confusion Matrices. (**a**) 3-layer CNN, (**b**) 4-layer CNN, (**c**) 5-layer CNN, and (**d**) 6-layer CNN.

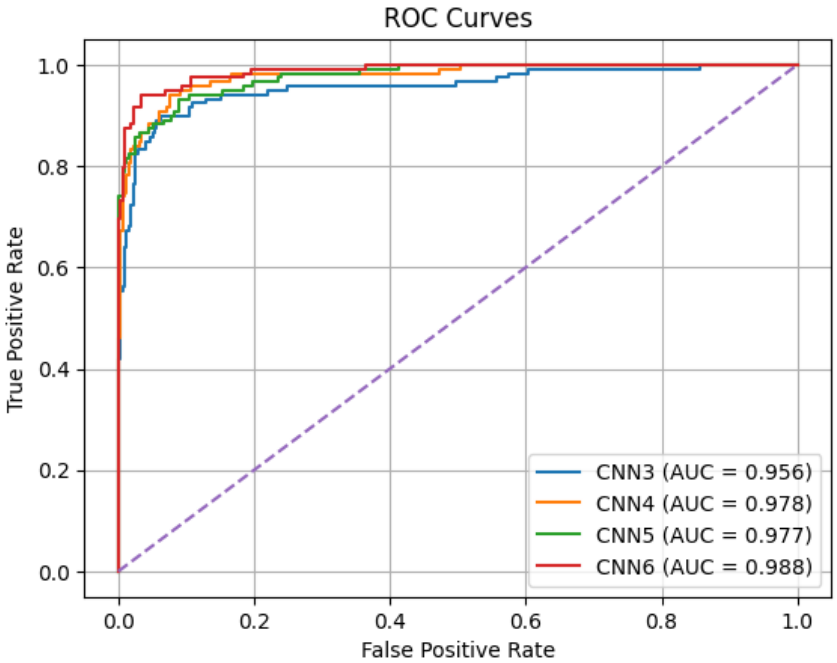

Figure 6. ROC curves of 3-layer CNN, 4-layer CNN, 5-layer CNN, and 6-layer CNN models. The Dash Line represents the 50% accuracy performance of models.

According to Figure 6, it can be observed that the standalone models present an AUC above 95%, with CNN6 being the best model at an AUC of 98.80%, followed closely by CNN5 and CNN4. CNN3 is the model with the lowest AUC (95.60%).

With the aim of improving the results, ensemble models with varying weights were implemented. The results of these models are shown in Table 6. The best ensemble model is the one that combines the four aforementioned models, presenting the confusion matrix shown in Figure 7.

Table 6. Results of standalone and ensemble models.

Model	Accuracy	Precision	Recall	F1-Score
CNN3	0.9202	0.7857	**0.9167**	0.8462
CNN4	0.9302	0.8929	0.8475	0.8696
CNN5	0.9235	0.9107	0.8160	0.8608
CNN6	0.9465	**0.9554**	0.8560	0.9029
CNN3 + CNN4 + CNN5	0.9326	0.8929	0.8547	0.8734
CNN4 + CNN5 + CNN6	0.9465	0.9375	0.8678	0.9013
CNN3 + CNN4 + CNN5 + CNN6	**0.9488**	0.9375	0.8750	**0.9052**

Values in bold indicate the best value obtained for the metric.

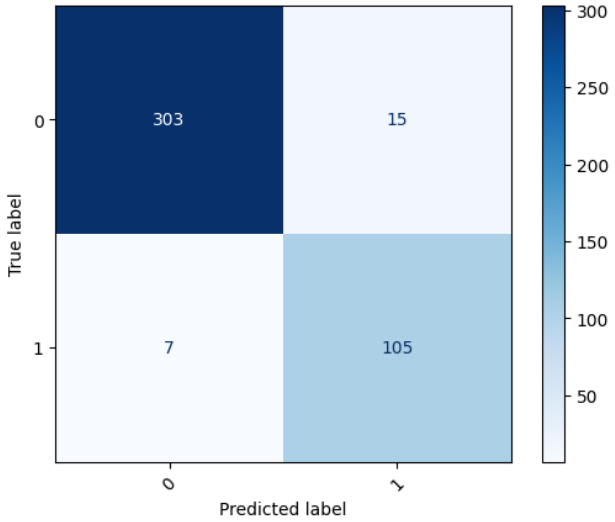

Figure 7. Confusion matrix of the weighted ensemble model based on CNN3, CNN4, CNN5, and CNN6.

According to Table 6, it can be observed that in most metrics, such as Accuracy, Precision, and F1-Score, the best standalone model is the CNN model with 6 layers (CNN6). However, in terms of Recall, the best model is the CNN with 3 layers (CNN3).

On the other hand, it can be observed that of the two weighted ensemble models based on three models, the CNN4 + CNN5 + CNN6 model outperforms the best standalone CNN6 model in terms of Accuracy and Recall. However, in terms of Precision and F1-Score, it yields lower results.

The weighted ensemble model based on the four standalone models achieves the best results, outperforming the best standalone model in Accuracy, Recall, and F1-Score.

To define the ensemble weights, the standalone models were first ranked according to their individual F1-Score values, since this metric provides a balanced evaluation between precision and recall and is more appropriate for imbalanced classification scenarios. Based on this ranking, CNN6 obtained the highest performance, followed by CNN4, CNN5, and CNN3.

The weight assignment strategy was designed under the assumption that models with higher predictive reliability should contribute more strongly to the final ensemble

decision. Therefore, larger weights were assigned to the best-performing models, while lower-performing models received smaller contributions.

Additionally, several alternative weight distributions were experimentally evaluated for both the three-model and four-model ensemble configurations in order to analyze the sensitivity of the ensemble to different contribution levels. The evaluated combinations are presented in Table 7. Among them, the second weight configuration achieved the best validation performance and was therefore selected for the final ensemble model.

Table 7. The weights used to create the ensemble models.

Number of Models	Weight Set		
	1	**2**	**3**
3	Same weight	0.40, 0.35, and 0.25	0.50, 0.33, and 0.17
4	Same weight	0.40, 0.30, 0.20, and 0.10	0.50, 0.30, 0.15, and 0.05

The ensemble prediction was computed as a weighted average of the individual model probabilities according to Equation (6).

$$P_{ensemble} = \sum_{i=1}^{n} w_i P_i \quad (6)$$

where

P_i: Predicted probability by each model,
w_i: Weight assigned.

$$\sum_{i=1}^{n} w_i = 1$$

Higher weights were assigned to models with higher F1-Score values in order to prioritize models with better balance between false positives and false negatives.

4.2. Discussions

In this section, the results achieved by the weighted ensemble model are discussed, and these are compared with the results achieved by other state-of-the-art models.

4.2.1. Benchmark Models

To analyze the results of the proposed model, three well-known state-of-the-art models were implemented: VGG-16, ResNet18, and MobileNetV2, whose architectures are shown in Table 8.

Table 8. Hyperparameters of the benchmark models.

CNN Model	Input Size	Number of Layers	Trainable Parameters
VGG-16	224 × 224 × 3	16	138 M
ResNet-18	224 × 224 × 3	18	11 M
MobileNetV2	224 × 224 × 3	53	3.49 M

VGG-16 is a well-known convolutional neural network (CNN) architecture in computer vision that contains 16 convolutional layers and 3 fully connected layers.

ResNet-18 is another convolutional neural network (CNN) architecture that belongs to the ResNet (Residual Networks) family. It has 18 deep layers, primarily CNNs, and a final fully connected layer. ResNet uses skip connections, or residual connections, that allow information to bypass layers.

MobileNet is a family of convolutional neural networks (CNNs) designed by Google to be used on resource-constrained devices such as mobile phones, tablets, or embedded systems.

Table 8 shows the hyperparameters of the benchmark models.

Table 9 shows the results achieved by the benchmark models.

Table 9. Results of the benchmark models.

Model	Accuracy	Precision	Recall	F1-Score
VGG-16	0.9209	0.7321	**0.9535**	0.8283
ResNet-18	0.8372	0.8571	0.6400	0.7328
MobileNetV2	0.8791	0.6339	0.8659	0.7320
Proposal	**0.9488**	**0.9375**	0.8750	**0.9052**

Values in bold indicate the best value obtained for the metric.

According to Table 9, in general, the three benchmark models yield lower results than the weighted ensemble model. The benchmark model that stands out most among the three is VGG-16, with a Recall (0.9535) that surpasses all other models, including the one proposed in this study. In terms of Accuracy (0.9488), Precision (0.9375), and F1-Score (0.9052), the weighted ensemble model proposed in this study achieves superior results compared to all benchmark models.

It is important to highlight that the simpler models outperform the larger models in most metrics. This does not mean they are generally better, but rather that for the specific problem of anthracnose detection in avocado fruits, classical CNNs and the weighted ensemble model are more suitable.

It is also worth noting that the avocado domain is not represented in the ImageNet dataset, and that the large models have been trained on this collection.

Also, it is important to highlight the superiority of VGG-16 regarding the Recall metric. According to Equation (4), Recall is based on TP and FN; for the VGG-16 model, a TP of 82 and an FN of 4 were obtained. Although this TP is not better than the TP of 105 achieved by the proposed model, the FN of 4 is superior to the proposed model's FN of 15, which greatly influences the Recall estimation. The main strength of VGG-16 compared to all implemented models is its TN prediction of 314, outperforming all other models. This directly influences the FN prediction, presenting the lowest value among them.

4.2.2. Statistical Analysis

To determine whether a statistically significant difference exists between the proposed model and the benchmark models, the McNemar test was conducted, the results of which are shown in Table 10.

Table 10. *p*-values for McNemar test between proposal and benchmark models.

Model	*p*-Value
VGG-16	0.1641
ResNet-18	3.91×10^{-10}
MobileNetV2	4.71×10^{-5}

According to Table 10, the McNemar test reveals that the differences between the proposed model and both ResNet-18 and MobileNetV2 are statistically significant, since their *p*-values are lower than 0.05. This indicates that the prediction errors produced by these benchmark models differ significantly from those of the proposed approach, suggesting that the proposed model achieves a genuinely different and improved classification behavior rather than a performance variation caused by random fluctuations.

In contrast, the *p*-value obtained for VGG-16 is greater than 0.05, indicating that the differences between the proposed model and VGG-16 are not statistically significant. Although the proposed approach achieved slightly better performance metrics, the McNemar test suggests that both models exhibit comparable classification behavior on the evaluated dataset.

Overall, these results support the robustness of the proposed model and confirm that its performance improvement over ResNet-18 and MobileNetV2 is statistically reliable.

4.2.3. Computational Cost and Latency

The training time of the implemented models, as well as the inference time in seconds on a computer with a HP Boeblingen, Germany, Core i7-13700H, processor at 2.40 GHz, 16 GB of RAM, running on a Windows 11 environment, is shown in Table 11.

Table 11. Training time and latency in seconds for the implemented models.

Model	Computational Cost	Latency	
		Test Set	per Image
CNN3	14,357	112	0.26
CNN4	12,382	126	0.29
CNN5	16,734	130	0.30
CNN6	15,165	122	0.28
VGG-16	12,155	194	0.59
ResNet-18	10,260	142	0.43
MobileNetV2	8540	142	0.43

In this study, the computational cost refers to the time required to train a model and is estimated through Equation (7).

$$T = \sum_{e=1}^{E} \left(\sum_{s=1}^{S} (t_s + t_{val}) \right) \tag{7}$$

where

T: Total training time in seconds.
E: Total number of epochs executed during the training phase.
S: Number of steps (batches) per epoch.
t_s: Inference and backpropagation time for a single step s.
t_{val}: Time taken for evaluation on the validation set at the end of each epoch.

According to Table 11, the benchmark models based on transfer learning exhibit lower training times compared to the CNN models trained from scratch. This behavior can be explained by the use of pre-trained ImageNet weights in VGG-16, ResNet-18, and MobileNetV2, where only the final classification layers are fine-tuned during training. As a result, the optimization process converges faster and requires fewer computational resources than models initialized with random weights. In contrast, the CNN3–CNN6 models must learn feature representations entirely from the training dataset, which increases the computational cost and training time.

Regarding inference latency, ResNet-18 and MobileNetV2 achieved similar performance, requiring 142 s for the complete test set and approximately 0.43 s per image. In comparison, VGG-16 presented the highest latency (194 s and 0.59 s per image), mainly due to its large number of parameters, particularly in the fully connected layers located at the end of the architecture. These dense layers significantly increase the computational burden during inference.

The CNN models trained from scratch (CNN3–CNN6) showed lower inference latency than the benchmark models, with per-image processing times ranging from 0.26 s to

0.30 s. This reduction is associated with their simpler architectures and lower parameter counts, which decrease computational complexity during forward propagation. Among these models, CNN3 achieved the lowest latency, whereas CNN5 required the highest computational cost among the proposed CNN architectures.

Finally, although the ensemble model achieved the best classification performance, its inference latency is inherently higher because predictions from multiple standalone models must be computed and combined. Consequently, the total latency of the ensemble corresponds approximately to the accumulated latency of its constituent models, representing a trade-off between predictive accuracy and computational efficiency.

4.2.4. Extra Tests

Additionally, beyond the experiments reported in this study, the proposed model was evaluated on two additional datasets: one containing data collected in 2026, and another corresponding to the dataset used to train the models reported in [11].

The 2026 dataset

This dataset is still under development. At present, it consists of 233 images, including 154 images belonging to class 0 (healthy) and 79 images belonging to class 1 (unhealthy). These images were collected between March and May 2026. The same imaging equipment used for the 2025 dataset was employed for image acquisition.

Figure 8 presents the results obtained by the proposed model on the 2026 dataset, which was not used during training. The confusion matrix indicates an accuracy of 0.9657, a precision of 0.8987, a recall of 1.0000, and an F1-score of 0.9467. Despite being collected in a different year, the dataset yielded performance metrics comparable to those obtained on the original test set, demonstrating the robustness of the proposed model and its ability to generalize across different data acquisition periods.

The dataset used in [11].

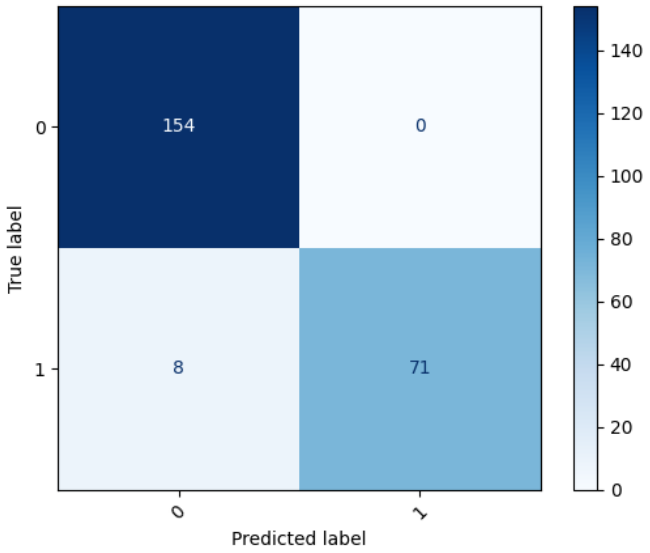

Figure 8. Confusion matrix of the weighted ensemble model for the 2026 dataset.

This dataset contains three classes: healthy, anthracnose, and scab. For the purposes of this study, only the first two classes were considered, resulting in a total of 466 images, including 283 images from class 0 (healthy) and 183 images from class 1 (unhealthy). The image resolution is 300 × 300 pixels, which is considerably lower than that of the images used in the previous datasets. Figure 9 presents the results obtained on this dataset.

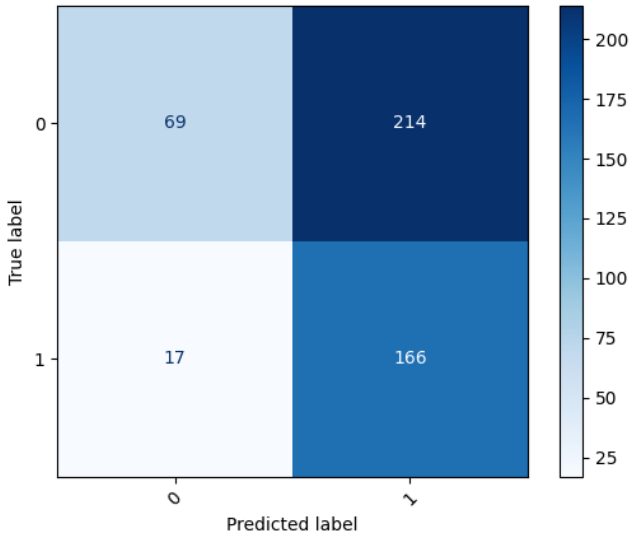

Figure 9. Confusion matrix of the weighted ensemble model for the dataset used in [11].

According to Figure 9, when this dataset was used as an additional test set, the proposed model achieved an accuracy of 0.5043, a precision of 0.9071, a recall of 0.4368, and an F1-score of 0.5897. These results indicate that the model does not generalize well to this dataset. In particular, the confusion matrix reveals a high number of false negatives, resulting in a substantial reduction in recall. This performance degradation is likely attributable to the significant differences between this dataset and the datasets used during training. As summarized in Table 12, factors such as image resolution, image shape, color characteristics, acquisition conditions, and dataset composition may have contributed to the observed decline in predictive performance.

Table 12. Differences between our datasets and the dataset in [11].

Factor	Our Datasets	Dataset in [11]
Image size	3072 × 480	300 × 300
Image shape	Rectangular	Square
Background	Gray	White
Color tone	Dark green	Varied
		

These results suggest the presence of a domain shift between the training dataset and this external dataset, which limits the model's ability to generalize effectively. The main factors that may explain this behavior are summarized in Table 12.

5. Conclusions and Future Work

5.1. Conclusions

Based on the experimental results, the proposed weighted ensemble model demonstrated strong performance for anthracnose detection in Fuerte avocado fruits, achieving an F1-score of 0.9052, a precision of 0.9375, and an accuracy of 0.9488. These results indicate that the proposed approach is an effective and computationally efficient alternative for automated anthracnose detection.

Furthermore, the weighted ensemble consistently achieved better overall performance than the benchmark models across most evaluation metrics. However, the statistical analy-

sis performed using McNemar's test showed that the observed improvements were statistically significant only when compared with ResNet-18 and MobileNetV2. Therefore, while the proposed model exhibited superior predictive performance, the differences with some competing approaches should be interpreted with caution from a statistical perspective.

Overall, the findings suggest that combining multiple shallow CNN architectures through a weighted ensemble strategy can provide a robust solution for anthracnose detection, while maintaining lower computational complexity than deeper state-of-the-art models.

5.2. Future Work

Although the proposed weighted ensemble achieved the best overall performance, the results indicate several opportunities for further improvement. In particular, the recall value obtained by the proposed model (0.8750) remains lower than that achieved by VGG-16 (0.9535). Since recall is especially important in disease detection applications, where minimizing false negatives is critical, future research could investigate hybrid ensemble or stacking approaches that combine lightweight CNN architectures with high-performing deep models to further enhance disease identification capabilities.

In addition, the experimental results showed that the CNN6 model achieved performance comparable to that of the weighted ensemble while requiring a simpler architecture. Therefore, future work may focus on optimizing individual CNN models through decision-threshold tuning, probability calibration, and advanced hyperparameter optimization techniques. Such strategies could improve the balance between true positive and true negative predictions while reducing computational requirements and inference latency.

Finally, future studies should evaluate the proposed approach on larger and more diverse datasets collected under different environmental conditions, acquisition devices, and geographical locations. This would enable a more comprehensive assessment of the model's robustness, generalization capability, and suitability for deployment in real-world agricultural scenarios.

Author Contributions: Conceptualization, A.F., S.H. and H.T.-C.; methodology, A.F., J.G.-V. and R.M.-G.; software, J.G.-V.; validation, E.F.-Q., C.S.-D. and H.T.-C.; formal analysis, A.F. and S.H.; investigation, A.F. and H.T.-C.; resources, C.S.-D. and E.F.-Q.; data curation, J.G.-V. and R.M.-G.; writing—original draft preparation, A.F. and S.H.; writing—review and editing, H.T.-C., E.F.-Q. and C.S.-D.; visualization, A.F.; supervision, A.F.; project administration, C.S.-D.; funding acquisition, E.F.-Q. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data used in this study can be requested from the corresponding author.

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

References

1. Bravo-Pérez, D.; Santillán-Galicia, M.T.; Johansen-Naime, R.; González-Hernández, H.; Segura-León, O.; Ochoa-Martínez, D.; Guzman-Valencia, S. Species Diversity of Thrips (Thysanoptera) in Selected Avocado Orchards from Mexico Based on Morphology and Molecular Data. *J. Integr. Agric.* **2018**, *17*, 2509–2517. [CrossRef]
2. Guerrero-Barajas, C.; Constantino-Salinas, E.A.; Amora-Lazcano, E.; Tlalapango-Ángeles, D.; Mendoza-Figueroa, J.S.; Cruz-Maya, J.A.; Jan-Roblero, J. *Bacillus mycoides* A1 and *Bacillus tequilensis* A3 Inhibit the Growth of a Member of the Phytopathogen *Colletotrichum gloeosporioides* Species Complex in Avocado. *J. Sci. Food Agric.* **2020**, *100*, 4049–4056. [CrossRef]
3. Thanh, L.T.H.; Tien, N.T.T.; Trang, N.T.T.; Trieu, T.Q.; Tien, N.Q.D.; Loc, N.H. First Report of *Colletotrichum siamense* and *Colletotrichum endophyticum* Associated with Anthracnose on Avocado (*Persea americana* Mill.) in Vietnam. *J. Plant Pathol.* **2025**, *107*, 633–647. [CrossRef]

4. Tapia, L.; Castillo-Novales, D.; Riquelme, N.; Valencia, A.L.; Larach, A.; Cautín, R.; Besoain, X. Rainfall and High Humidity Influence the Seasonal Dynamics of Spores of *Glomerellaceae* and *Botryosphaeriaceae* Genera in Avocado Orchards and Their Fruit Rot Association. *Agronomy* **2025**, *15*, 1453. [CrossRef]
5. Thangaraj, R.; Dinesh, D.; Hariharan, S.; Rajendar, S.; Gokul, D.; Hariskarthi, T.R. Automatic Recognition of Avocado Fruit Diseases Using Modified Deep Convolutional Neural Network. *Int. J. Grid Distrib. Comput.* **2020**, *13*, 1550–1559.
6. Torne, S.; Shetty, D.K.; Makkithaya, K.; Hegde, P.; Sudhi, M.; Kumar Pullela, P.; Tamil Eniyan, T.; Kamath, R.; Salu, S.; Bhat, P.; et al. VGG-16, VGG-16 with Random Forest, Resnet50 with SVM, and EfficientNetB0 with XGBoost-Enhancing Bone Fracture Classification in X-Ray Using Deep Learning Models. *IEEE Access* **2025**, *13*, 25568–25577. [CrossRef]
7. Thongpance, N.; Dangyai, P.; Roongprasert, K.; Wongkamhang, A.; Saosuwan, R.; Chotikunnan, R.; Imura, P.; Nirapai, A.; Chotikunnan, P.; Sangworasil, M.; et al. Exploring ResNet-18 Estimation Design through Multiple Implementation Iterations and Techniques in Legacy Databases. *J. Robot. Control* **2023**, *4*, 650–661. [CrossRef]
8. Gulzar, Y. Fruit Image Classification Model Based on MobileNetV2 with Deep Transfer Learning Technique. *Sustainability* **2023**, *15*, 1906. [CrossRef]
9. Matsui, T.; Sugimori, H.; Koseki, S.; Koyama, K. Automated Detection of Internal Fruit Rot in Hass Avocado via Deep Learning-Based Semantic Segmentation of X-Ray Images. *Postharvest Biol. Technol.* **2023**, *203*, 112390. [CrossRef]
10. Chauhan, A.; de Villiers, H.A.C.; Meesters, L.; Paillart, M.J.M.; Grbović, Ž.; Panić, M.; Brdar, S. Avocado Stem-End Rot Detection Using Hyperspectral Imaging. *Acta Hortic.* **2024**, *1396*, 107–114. [CrossRef]
11. Moreano, M.; Sosa, A.; Mauricio, D.; Rivera, L.; Santisteban, J. ApaltAI: A Web-Based Diagnostic System with a Sequential Voting Architecture for Detecting Anthracnose and Scab in Avocado Fruit. *Front. Plant Sci.* **2026**, *17*, 1736123. [CrossRef]
12. De Castro, A.I.; Ehsani, R.; Ploetz, R.; Crane, J.H.; Abdulridha, J. Optimum Spectral and Geometric Parameters for Early Detection of Laurel Wilt Disease in Avocado. *Remote Sens. Environ.* **2015**, *171*, 33–44. [CrossRef]
13. De Castro, A.I.; Ehsani, R.; Ploetz, R.C.; Crane, J.H.; Buchanon, S. Detection of Laurel Wilt Disease in Avocado Using Low Altitude Aerial Imaging. *PLoS ONE* **2015**, *10*, e0124642. [CrossRef]
14. Shuaibu, M.; Lee, W.S.; Schueller, J.; Gader, P.; Hong, Y.K.; Kim, S. Unsupervised Hyperspectral Band Selection for Apple Marssonina Blotch Detection. *Comput. Electron. Agric.* **2018**, *148*, 45–53. [CrossRef]
15. Soini, C.T.; Fellah, S.; Abid, M.R. Citrus Greening Infection Detection (CIGID) by Computer Vision and Deep Learning. In *Proceedings of the 2019 3rd International Conference on Information System and Data Mining (ICISDM'19)*; ACM International Conference Proceeding Series; Association for Computing Machinery: New York, NY, USA, 2019. [CrossRef]
16. Hariharan, J.; Fuller, J.; Ampatzidis, Y.; Abdulridha, J.; Lerwill, A. Finite Difference Analysis and Bivariate Correlation of Hyperspectral Data for Detecting Laurel Wilt Disease and Nutritional Deficiency in Avocado. *Remote Sens.* **2019**, *11*, 1748. [CrossRef]
17. Abdulridha, J.; Ehsani, R.; Abd-Elrahman, A.; Ampatzidis, Y. A Remote Sensing Technique for Detecting Laurel Wilt Disease in Avocado in Presence of Other Biotic and Abiotic Stresses. *Comput. Electron. Agric.* **2019**, *156*, 549–557. [CrossRef]
18. Logashov, D.; Shadrin, D.; Somov, A.; Pukalchik, M.; Uryasheva, A.; Gupta, H.P.; Rodichenko, N. Apple Trees Diseases Detection through Computer Vision in Embedded Systems. In *IEEE International Symposium on Industrial Electronics*; IEEE: New York, NY, USA, 2021. [CrossRef]
19. Mahmud, M.S.; Zahid, A.; He, L.; Martin, P. Opportunities and Possibilities of Developing an Advanced Precision Spraying System for Tree Fruits. *Sensors* **2021**, *21*, 3262. [CrossRef]
20. Li, H.; Jin, Y.; Zhong, J.; Zhao, R. A Fruit Tree Disease Diagnosis Model Based on Stacking Ensemble Learning. *Complexity* **2021**, *2021*, 6868592. [CrossRef]
21. Anagnostis, A.; Tagarakis, A.C.; Asiminari, G.; Papageorgiou, E.; Kateris, D.; Moshou, D.; Bochtis, D. A Deep Learning Approach for Anthracnose Infected Trees Classification in Walnut Orchards. *Comput. Electron. Agric.* **2021**, *182*, 105998. [CrossRef]
22. Moriya, É.A.S.; Imai, N.N.; Tommaselli, A.M.G.; Berveglieri, A.; Santos, G.H.; Soares, M.A.; Marino, M.; Reis, T.T. Detection and Mapping of Trees Infected with Citrus Gummosis Using UAV Hyperspectral Data. *Comput. Electron. Agric.* **2021**, *188*, 106298. [CrossRef]
23. Javidan, S.M.; Banakar, A.; Vakilian, K.A.; Ampatzidis, Y. A Feature Selection Method Using Slime Mould Optimization Algorithm in Order to Diagnose Plant Leaf Diseases. In *2022 8th International Iranian Conference on Signal Processing and Intelligent Systems, ICSPIS 2022*; IEEE: New York, NY, USA, 2022. [CrossRef]
24. Savla, D.; Dhaka, V.S.; Rani, G.; Oza, M. Apple Leaf Disease Detection and Classification Using CNN Models. In *Smart Innovation, Systems and Technologies*; Springer Nature: Singapore, 2022; Volume 303 SIST. [CrossRef]
25. Chaschatzis, C.; Karaiskou, C.; Mouratidis, E.G.; Karagiannis, E.; Sarigiannidis, P.G. Detection and Characterization of Stressed Sweet Cherry Tissues Using Machine Learning. *Drones* **2022**, *6*, 3. [CrossRef]
26. Genangeli, A.; Allasia, G.; Bindi, M.; Cantini, C.; Cavaliere, A.; Genesio, L.; Giannotta, G.; Miglietta, F.; Gioli, B. A Novel Hyperspectral Method to Detect Moldy Core in Apple Fruits. *Sensors* **2022**, *22*, 4479. [CrossRef]

27. Liu, B.Y.; Fan, K.J.; Su, W.H.; Peng, Y. Two-Stage Convolutional Neural Networks for Diagnosing the Severity of Alternaria Leaf Blotch Disease of the Apple Tree. *Remote Sens.* **2022**, *14*, 2519. [CrossRef]
28. Cui, R.; Li, J.M.; Wang, Y.; Fang, S.; Yu, K.; Zhao, Y. Hyperspectral Imaging Coupled with Dual-Channel Convolutional Neural Network for Early Detection of Apple Valsa Canker. *Comput. Electron. Agric.* **2022**, *202*, 107411. [CrossRef]
29. Xiao, D.; Pan, Y.; Feng, J.; Yin, J.; Liu, Y.; He, L. Remote Sensing Detection Algorithm for Apple Fire Blight Based on UAV Multispectral Image. *Comput. Electron. Agric.* **2022**, *199*, 107137. [CrossRef]
30. Sandhya, S.; Balasundaram, A.; Sivaraman, A. Deep Learning and Computer Vision Based Model for Detection of Diseased Mango Leaves. *Int. J. Recent Innov. Trends Comput. Commun.* **2022**, *10*, 70–79. [CrossRef]
31. Gomez-Flores, W.; Garza-Saldana, J.J.; Varela-Fuentes, S.E. A Huanglongbing Detection Method for Orange Trees Based on Deep Neural Networks and Transfer Learning. *IEEE Access* **2022**, *10*, 116686–116696. [CrossRef]
32. Zhang, Q.; Huang, W.; Wang, Q.; Wu, J.; Li, J. Detection of Pears with Moldy Core Using Online Full-Transmittance Spectroscopy Combined with Supervised Classifier Comparison and Variable Optimization. *Comput. Electron. Agric.* **2022**, *200*, 107231. [CrossRef]
33. Nandhini, M.; Kala, K.U.; Thangadarshini, M.; Madhusudhana Verma, S. Deep Learning Model of Sequential Image Classifier for Crop Disease Detection in Plantain Tree Cultivation. *Comput. Electron. Agric.* **2022**, *197*, 106915. [CrossRef]
34. Zhang, S.; Li, X.; Ba, Y.; Lyu, X.; Zhang, M.; Li, M. Banana Fusarium Wilt Disease Detection by Supervised and Unsupervised Methods from UAV-Based Multispectral Imagery. *Remote Sens.* **2022**, *14*, 1231. [CrossRef]
35. BaniMustafa, A.; Qattous, H.; Ghabeish, I.; Karajeh, M. A Machine Learning Hybrid Approach for Diagnosing Plants Bacterial and Fungal Diseases. *Int. J. Adv. Comput. Sci. Appl.* **2023**, *14*, 912–921. [CrossRef]
36. Yang, Y.; He, L.; Peter, K.A. Smartphone-Assisted Apple Scab Identification and Quantification Using Artificial Intelligence. In *2023 ASABE Annual International Meeting*; American Society of Agricultural and Biological Engineers: St. Joseph, MI, USA, 2023. [CrossRef]
37. Biswas, B.; Yadav, R.K. Multilayer Convolutional Neural Network Based Approach to Detect Apple Foliar Disease. In *2023 2nd International Conference for Innovation in Technology, INOCON 2023*; IEEE: New York, NY, USA, 2023. [CrossRef]
38. da Silva, J.C.F.; Silva, M.C.; Luz, E.J.S.; Delabrida, S.; Oliveira, R.A.R. Using Mobile Edge AI to Detect and Map Diseases in Citrus Orchards. *Sensors* **2023**, *23*, 2165. [CrossRef]
39. Win Kent, O.; Weng Chun, T.; Lee Choo, T.; Weng Kin, L. Early Symptom Detection of Basal Stem Rot Disease in Oil Palm Trees Using a Deep Learning Approach on UAV Images. *Comput. Electron. Agric.* **2023**, *213*, 108192. [CrossRef]
40. Hariharan, J.; Ampatzidis, Y.; Abdulridha, J.; Batuman, O. An AI-Based Spectral Data Analysis Process for Recognizing Unique Plant Biomarkers and Disease Features. *Comput. Electron. Agric.* **2023**, *204*, 107574. [CrossRef]