applied sciences

Article

Automatic Detection of Tomato Diseases Using Deep Transfer Learning

Natheer Khasawneh [1,*], Esraa Faouri [2] and Mohammad Fraiwan [2]

[1] Department of Software Engineering, Jordan University of Science and Technology, Irbid 22110, Jordan
[2] Department of Computer Engineering, Jordan University of Science and Technology, Irbid 22110, Jordan
* Correspondence: natheer@just.edu.jo

Abstract: Global food production is being strained by extreme weather conditions, fluctuating temperatures, and geopolitics. Tomato is a staple agricultural product with tens of millions of tons produced every year worldwide. Thus, preserving the tomato plant from diseases will go a long way in reducing economical loss and boost output. Technological innovations have great potential in facilitating disease detection and control. More specifically, artificial intelligence algorithms in the form of deep learning methods have established themselves in many real-life applications in a wide range of disciplines (e.g., medicine, agriculture, or facial recognition, etc.). In this paper, we aim at applying deep transfer learning in the classification of nine tomato diseases (i.e., bacterial spot, early blight, late blight, leaf mold, mosaic virus, septoria leaf spot, spider mites, target spot, and yellow leaf curl virus) in addition to the healthy state. The approach in this work uses leaf images as input, which is fed to convolutional neural network models. No preprocessing, feature extraction, or image processing is required. Moreover, the models are based on transfer learning of well-established deep learning networks. The performance was extensively evaluated using multiple strategies for data split and a number of metrics. In addition, the experiments were repeated 10 times to account for randomness. The ten categories were classified with mean values of 99.3% precision, 99.2% F1 score, 99.1% recall, and 99.4% accuracy. Such results show that it is highly feasible to develop smartphone-based applications that can aid plant pathologists and farmers to quickly and accurately perform disease detection and subsequent control.

Keywords: deep learning; tomatoes; virus; bacteria; blight; spot; mold; image classification; artificial intelligence

check for updates

Citation: Khasawneh, N.; Faouri, E.; Fraiwan, M. Automatic Detection of Tomato Diseases Using Deep Transfer Learning. *Appl. Sci.* **2022**, *12*, 8467. https://doi.org/10.3390/app12178467

Academic Editor: Ángel González-Prieto

Received: 21 July 2022
Accepted: 23 August 2022
Published: 24 August 2022

1. Introduction

Agriculture is one of humanity's most critical activities, of which plant disease control is a cornerstone. It is necessary to pay attention to the quality and wellbeing of the agricultural harvest. This will help maintain food production levels in the face of natural diseases and aid countries in coping with political and environmental challenges. Tomatoes are among the vital crops and staple food products around the world because of their rich nutritional content and their role in many recipes [1]. The food and agriculture organization (FAO) ranks tomatoes as the sixth most abundant vegetable around the world [2]. In 2017, nearly 170.8 million tons of tomatoes were produced worldwide [3]. However, the tomato plant is susceptible to many diseases caused by bacteria, viruses, or fungi that have a direct adverse effect on productivity [4].

To detect plant diseases, farmers refer to plant pathologists. Alternatively, they can rely on their own experience or public resources. However, the required time, effort, and technical expertise may be prohibitive for most professional or hobby farmers [5]. Thus, technological solutions that can aid the disease detection and identification will go a long way in reducing cost and improving the accuracy and speed of disease control. In this regard, recent advances in artificial intelligence (AI) have empowered a wide swath of applications from various disciplines. AI systems capture domain knowledge in their models through the training and validation process. They provide decision-making capabilities with nontrivial sophistication and complexity [6,7]. More specifically, deep learning algorithms have enabled the capture of intricate relationships and features of real-life processes. Convolutional neural networks (CNNs) are one of the types of deep learning algorithms that were found to be particularly useful for direct image-based decision-making and objection detection [8].

Neural networks are comprised of three layers; input, output, and hidden. On the other hand, deep learning involves a far greater number of layers, which enables the capturing of input features and details at various scales. Out of the many deep learning artificial intelligence algorithms, convolutional neural networks are the most suitable for handling images as input [8]. Layers in a convolutional neural network perform a series of convolution operations using filters of various sizes, which is typically followed by a rectified linear unit (ReLU) activation function. The result from the ReLU is a feature map that is downsampled by the subsequent pooling layer. In general, the final layer before the output in CNN is a fully connected layer, which combines the various features learned from previous layers and feeds the output layer.

Building CNN models is an elaborate process, which needs to balance the computational cost with the ability to automatically extract appropriate features at various scales, orientations, colors, reflections, and spatial properties. Moreover, the models may suffer from overfitting, underfitting, or inefficiency. In addition, thorough evaluation is needed to establish the trustworthiness of the models. Luckily, several public and well-established models exist in the literature. These models offer a wide-range of reliable capabilities with great efficiency [9]. Importantly, these models can be reused via an approach called transfer learning. This method utilizes generically pre-trained models by reusing the network structure and retraining part or all of the models including the existing model weights and parameters. Harnessing these robust models accelerates the development of new innovative AI applications without reinventing new CNN architectures. This methodology was successfully employed in many AI solutions for image classification [6,7].

In the context of technological and AI-based innovations for tomato disease diagnosis, several studies were conducted in the literature. Mim et al. [10] developed a system that helps tomato farmers discover the type of disease using leaf images of the plant. The researchers used artificial intelligence algorithms and CNN to develop a six-class (five diseases and one healthy) classification model with an accuracy of 96.55%. Hlaing and Zaw [11] isolated the leaf image from the background, and used explicit feature extraction in the form of statistical properties and scale invariant feature transform of texture features. These descriptors fed a support vector machine (SVM) classifier, which distinguishes between seven input categories (six diseases and one healthy) with an accuracy of 84.7%. Kumar and Vani [12] experimented with four deep learning models: LeNet, VGG16, ResNet, and Xception for ten-class classification (nine diseases + one healthy) of tomato leaf images, and reported a maximum accuracy of 99.25% using VGG16. Similarly, Tm et al. [13] used the AlexNet, GoogleNet, and LeNet models for the same classification problem and achieved an accuracy range of 94–95%. Annabel and Muthulakshmi [14] used masking and threshold-based segmentation to identify and isolate infected areas of a leaf image. They extracted several features (e.g., dissimilarity, homogeneity, and contrast) and used a random forest classifier to category 3 diseases plus healthy leaves with an accuracy of 94.1%. Agarwal et al. [15] developed a custom CNN model by modifying the VGG16 structure. They compared this model with traditional machine learning models (e.g., random forest and decision trees) and three deep learning ones (i.e., VGG16, Inceptionv3, and MobileNet) for ten-class classification, and achieved an accuracy of 98.4%. Ouhami et al. [16] employed transfer learning of three models; DensNet-161, DensNet-121, and VGG16. The highest accuracy was achieved using DenseNet-161 (i.e., 95.65%). Similarly, Alhaj Ali et al. [17] used Inceptionv3 and reported the highest accuracy to be 99.8%.

However, the high aforementioned results were achieved with augmented images with duplication.

Other approaches were employed in the literature. In one avenue, deep learning algorithms were combined with traditional machine learning to solve the classification problem. Al-gaashani et al. [18] extracted features from leaf images using MobileNetv2 and NASNetMobile. After dimensionality reduction, the concatenation of these features fed into non-deep classification networks (i.e., random forest, SVM, multinomial logistical regression). In another methodology, deep object detection methods were applied on plant images to detect diseases on leaves. Liu and Wang [19] employed the you only look once version 3 (YOLOv3) algorithm to detect gray leaf spot disease. They reported a mean average precision of 92.5%. Similarly, Wang et al. [20] used Faster R-CNN and Mask R-CNN to detect eleven disease states (including healthy) in fruit images.

Other approaches were employed in the literature. In one avenue, deep learning algorithms were combined with traditional machine learning to solve the classification problem. Al-gaashani et al. [18] extracted features from leaf images using MobileNetv2 and NASNetMobile. After dimensionality reduction, the concatenation of these features fed into non-deep classification networks (i.e., random forest, SVM, multinomial logistical regression). In another methodology, deep object detection methods were applied on plant images to detect diseases on leaves. Liu and Wang [19] employed the you only look once version 3 (YOLOv3) algorithm to detect gray leaf spot disease. They reported a mean average precision of 92.5%. Similarly, Wang et al. [20] used Faster R-CNN and Mask R-CNN to detect eleven disease states (including healthy) in fruit images. Traditional methods were also used in the literature recently. Gadade and Kirange [21] extracted the features using Gabor filters, gray level co-occurrence matrix, and speeded up robust features. This approach involves less computational and memory overhead than deep learning, but it is less effective in solving the classification problem as demonstrated by their reported accuracy of 74%. Similarly, Lu et al. [22] presented spectral vegetation indices as features for classification using K-nearest neighbors (KNN), and they reported a 100% accuracy, albeit with a very small dataset (445 images).

This work is motivated by the following factors:

- The adoption and implementation of technological innovations is generally lacking in the agricultural literature in comparison with other fields (e.g., medicine). This is especially true for the number of artificial intelligence applications in agriculture versus in medicine.
- Traditional classification methods rely upon explicit feature extraction and/or image processing techniques, which may be sensitive to changes in image quality, orientation, size, lighting, noise, etc. Furthermore, the classification performance is directly affected by the quality of features on which it is based. Moreover, pre-processing increases the delay, computational requirements, and compounded errors. In addition, it may hinder the deployment of real-life applications if complicated actions are required by the user.
- Previous works suffer from several deficiencies. First, some of these studies artificially increase the size of the dataset by including subtle differences in the dataset images. However, deep learning models are known to be immune to such changes. This duplication artificially improves the results by exposing the model to recognizing the similarities with the original images rather than features of the disease or health states. Second, building a customized CNN model is fraught with risks in terms of overfitting, underfitting, efficiency, and hardware requirements. Using deep transfer learning with pre-existing network architecture, carries with it the inherent credibility of the thousands of applications based on these models and the extensive scrutiny they have gone through, although at the expense of perceived lack of novelty and originality. Third, transfer learning is able to achieve competitive if not superior performance.

In this paper, deep transfer learning was used to detect and classify tomato disease using images of infected leaves. This approach has the advantages of employing well-established, trustworthy, and robust models without the need to redesign/reinvent a custom architecture. Moreover, deep learning models can render feature extraction and image preprocessing needless. The contributions of this paper are as follows:

1. Develop deep transfer learning models for the detection and classification of tomato diseases from leaf images for nine tomato diseases: bacterial spot, early blight, late blight, leaf mold, mosaic virus, septoria leaf spot, spider mites, target spot, and yellow leaf curl virus. In addition, healthy leaves were discerned as a 10th class;
2. Implement transfer learning of eleven deep convolutional neural networks models for the classification of leaf images into ten classes. Future Implementation of such a system in smart devices will greatly help farmers do prompt disease control;
3. Evaluate the performance of the various models using multiple metrics that cover many aspects of the detection and classification capabilities. Moreover, the training and validation times were reported.

The remainder of this paper is organized as follows: the data, convolutional network models, and performance evaluation metrics and setup are presented in detail in Section 2, Section 3 discusses the performance evaluation results along with comparison to the related literature and discussion of the models, and we conclude in Section 4.

2. Materials and Methods

Figure 1 shows a diagram of all phases involved in the proposed approach. By using CNNs, performing explicit feature extraction is not required. Furthermore, there is no need for separating relevant image parts (i.e., segmentation). These steps and others are handled implicitly by the complex operations of the deep learning models. Given a generically pre-trained deep learning model, several changes need to be made to re-purpose the model to the specific application. First, replace the classification layer to match the number of classes in the application (i.e., 10 classes for this paper). Second, replace the learnable layer that combine features from previous layers with a new layer. This may be a fully connected layer or a convolution2d layer depending on the CNN model. Third, if training is to be made faster, then some initial layers can be frozen (i.e., layer weights will not be updated during training). The number of frozen layers can be determined empirically depending on the application and the resulting testing performance and training speed. No layers were frozen in this work as the available hardware permitted extensive training. Fourth, the dataset needs to be prepared by resizing the images to fit the CNN requirements (e.g., 256 $\times$ 256 to 224 $\times$ 224). Furthermore, the data are split into training and validation subsets. In addition, image augmentation operations may be performed to introduce more variety into the dataset and improve the learning process. Fifth, in this final step, the CNN network is retrained with the tomato dataset, and the performance is evaluated. The next few subsections give more details about each part.

2.1. Dataset

The dataset consists of 18,160 publicly available tomato leaf images displaying features of nine tomato diseases in addition to the healthy state. The number of images per class was as follows: 2127 bacterial spot, 1000 early blight, 1909 late blight, 952 leaf mold, 373 mosaic virus, 1771 septoria leaf spot, 1676 spider mites, 1404 target spot, 5357 yellow leaf curl virus, and 1591 healthy [23]. Each image represents a photo of a single leaf exhibiting one of the ten health classes. The photos were taken using a neutral background that appears somewhat unified for all images. In addition, each leaf appears at the center of each image. Although the images may contain irrelevant margins displaying the background, no cropping or pre-processing were performed. The public source of the images provided the dataset in JPEG format and a 256 $\times$ 256 resolution. Samples of leaf images of the nine diseases and healthy leaves are shown in Figure 2.

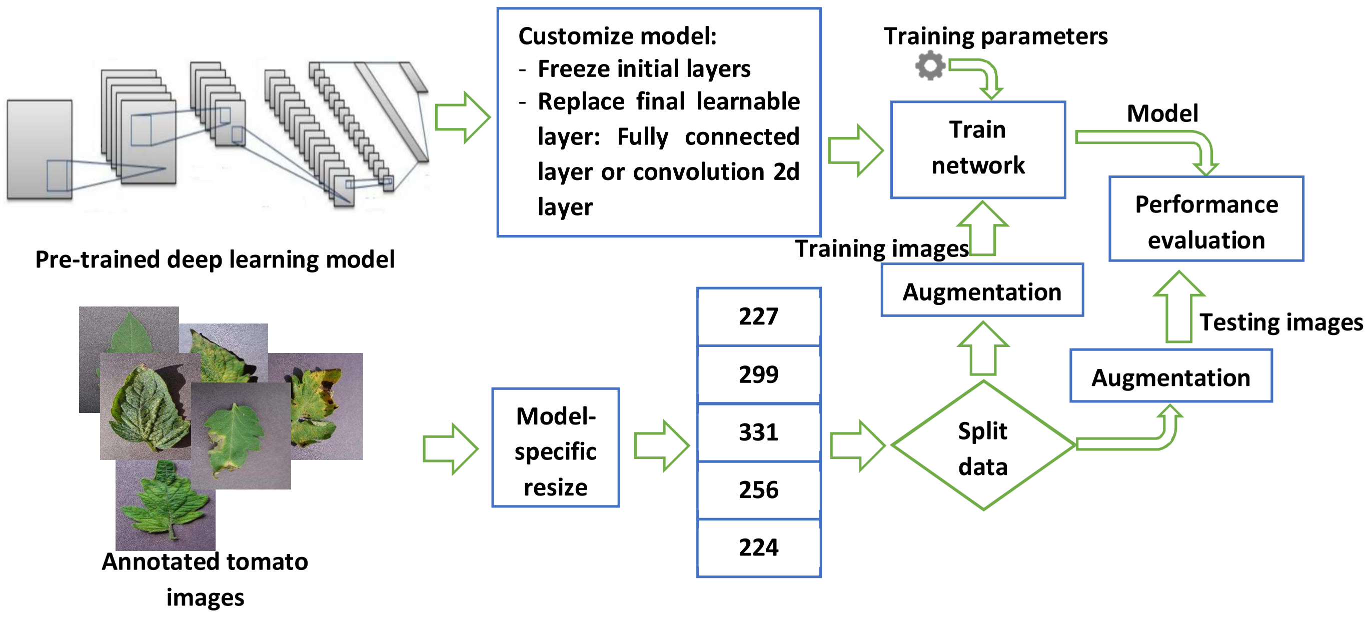


Figure 1. A diagram of all phases of the proposed approach.

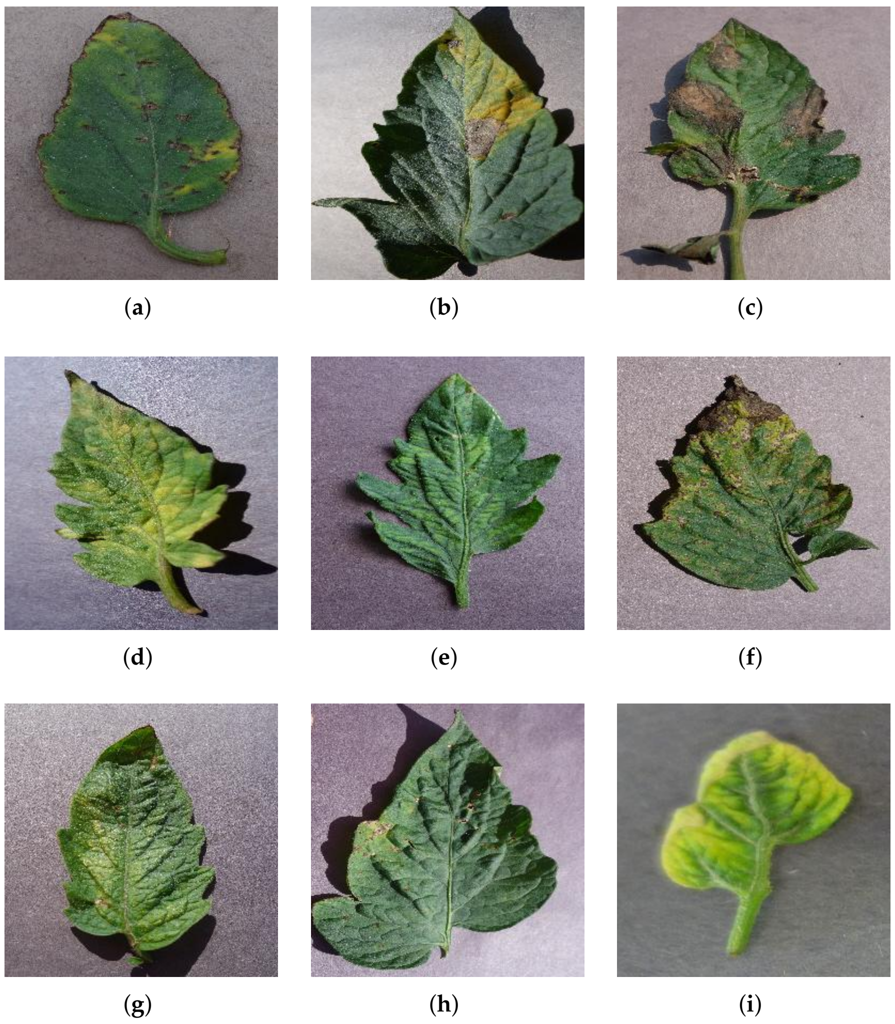


Figure 2. *Cont.*

(**j**)

Figure 2. Sample images from the nine disease classes and the healthy one. (**a**) bacterial spot; (**b**) early blight; (**c**) late blight; (**d**) leaf mold; (**e**) mosaic virus; (**f**) septoria leaf spot; (**g**) spider mites; (**h**) target spot; (**i**) yellow leaf curl virus; (**j**) healthy.

2.2. Convolutional Neural Network Models

This research investigated the customization, retraining, and use of eleven deep learning CNN models to classify tomato diseases from leaf images. These models differ in the input size, structure, and computational efficiency of their internal operations, among other differences. However, the hyperparameters (e.g., number of iterations), which fine-tuned the training of these models, were unified throughout this work. The CNN models were: DarkNet-53 [24], DenseNet-201 [25], GoogLeNet [26], Inceptionv3 [27], MobileNetv2, ResNet-18, ResNet-50, ResNet-101 [28], ShuffleNet [29], SqueezeNet [30], and Xception [31].

2.3. Performance Evaluation Setup

The training was performed using the same hyperparameters for all models. The number of training epochs was experimentally set to 5. This was based on the training and validation behavior of the models. Further training was deemed unnecessary. The available system memory allowed for a batch size of 16. The number of training iterations is equal to $\frac{No.\ input\ images \times No.\ of\ epochs}{batchsize} = \frac{5 \times No.\ input\ images}{16}$. The learning rate was set to 3×10^{-4}. The fast converging stochastic gradient descent with momentum (SGDM) was used as the solver optimization algorithm for network training [32].

Several data splitting strategies were used to test the models' ability to generalize to more data using a larger testing set (i.e., learn better with larger training set). The first strategy split the dataset into equal-sized training and validation sets (i.e., 50/50), the second one allocated 70% for training, and the last one used 90% of the images for training. Moreover, images in each set were augmented by performing scaling operations using random values from the range [0.9,1.1], and x–y translation using random values from the range [−30,30] pixels. In addition, random x-axis reflection (i.e., horizontal or vertical shifting of the image) was applied. Augmentation has been shown to improve the generalization of the learned knowledge [33]. It should be noted that augmentation did not increase the size of the dataset because the original images were discarded not duplicated.

The models were implemented and evaluated using MATLAB R2021a software running on an HP OMEN 30 L desktop GT13 with 64 GB RAM, NVIDIA® GeForce RTX™ 3080 GPU, Intel® Core™ i7-10700K CPU @ 3.80 GHz, and 1 TB SSD.

2.4. Performance Evaluation Metrics

The metrics used to evaluate the performance of the CNN models are shown in Equations (1)–(6). In these equations, T_P represents the true positive (i.e., a leaf image correctly classified in one of the nine disease states), F_N represents the false negative (i.e., a leaf image classified as healthy, but, in reality, it was drawn from one of the disease classes), F_P represents the false positive (i.e., a healthy leaf image wrongly classified as representing a disease), and T_N represents the true negative (i.e., a healthy image classified correctly as such). *Recall* (i.e., true positive rate (TPR) or sensitivity) measures the ability of the model to identify a leaf image as belonging to the correct disease class out of all the positive images, which is affected by the existence of false negatives. Moreover, *Specificity* measures the ability of the model to identify a leaf image as belonging to the healthy class, which is affected by the existence of false positives. High sensitivity indicates that the model easily recognizes leaf images as representing a disease but may include a large number of false positives. *Precision* measures the ratio of false positives to all cases identified as positive (i.e., false positives included). The *Accuracy* measures the ratio of the sum of true positives and true negatives to the total number of testing images. However, since different classes have a different number of images (i.e., class imbalance), the F1 score is considered a more reliable measure of the model classification performance [34]. The Matthews Correlation Coefficient (MCC), see Equation (6), is another metric of great importance. MCC and its multiclass generalization provide a more correct reflection of the classification performance in comparison to the accuracy and F1 score because the size imbalance of the different classes is taken under consideration [35]:

$$Accuracy = \frac{T_P + T_N}{P + N} \tag{1}$$

$$Precision = \frac{T_P}{T_P + F_P} \tag{2}$$

$$Recall = \frac{T_P}{T_P + F_N} \tag{3}$$

$$Specificity = \frac{T_N}{T_N + F_P} \tag{4}$$

$$F1 = 2 \times \frac{T_P}{2 \times T_P + F_P + F_N} \tag{5}$$

$$MCC = \frac{T_P \times T_N - F_P \times F_N}{\sqrt{(T_P + F_P)(T_P + F_N)(T_N + F_P)(T_N + F_N)}} \tag{6}$$

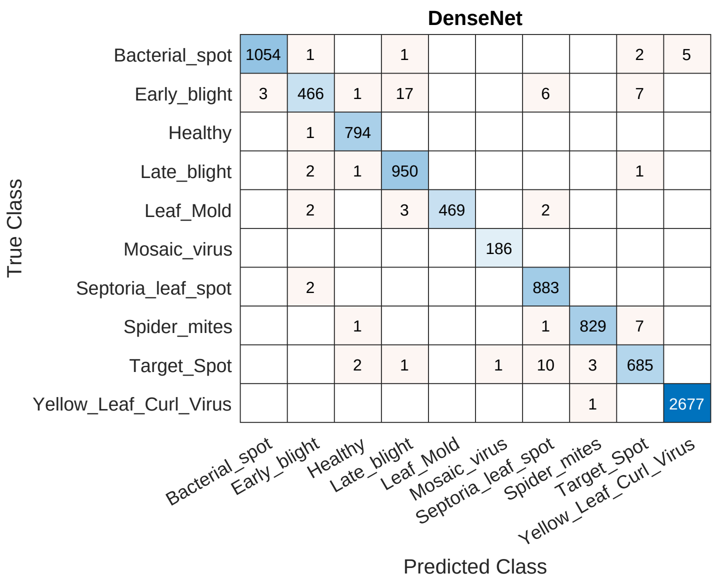
3. Results and Discussion

The performance evaluation was performed in order to gauge and compare the classification capabilities of the various deep transfer learning models using well-known and reflective performance indices. Moreover, the evaluation was repeated for 10 times to account for random choices for the various data subsets. In addition, the time requirements for training/validation were reported for all models under the various setups.

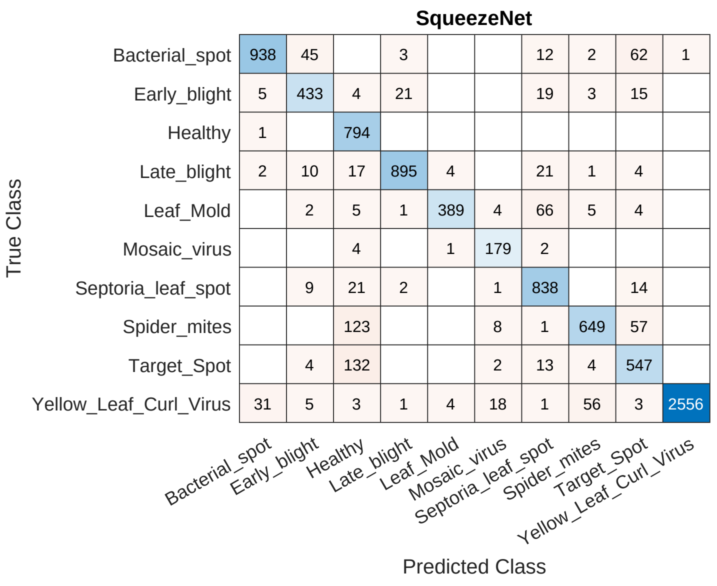
Three data split strategies were used (i.e., 50/50, 70/30, and 90/10), which may reveal the abilities of the different models in learning from more data, and any underfitting/overfitting anomalies. Table 1 shows the mean over 10 runs for the overall F1 score, precision, recall, specificity, and MCC using 50% of the data for training. Most models performed exceptionally well with the highest mean F1 score of 98.5% using DenseNet-201. The worst performing model was SqueezeNet with a 90.9% F1 score. These performance values are corroborated by the confusion matrices for the best and worst performing models as shown in Figure 3. The matrix for SqueezeNet shows a problematic trend of misclassifying leaves with diseases as healthy, especially spider mites and target spots.

Table 1. The mean overall F1 score, Precision, Recall, Specificity, and MCC using 50% of the data for training. The results are an average of 10 runs.

Model	F1 Score	Precision	Recall	Specificity	MCC
SqueezeNet	90.9%	91.9%	91.2%	99.2%	0.897
GoogLeNet	93.6%	93.7%	93.9%	99.5%	0.924
Inceptionv3	97.9%	98.0%	97.8%	99.8%	0.975
DenseNet-201	98.5%	98.6%	98.3%	99.9%	0.983
MobileNetv2	96.3%	96.5%	96.3%	99.7%	0.958
Resnet101	98.3%	98.3%	98.3%	99.9%	0.983
Resnet50	98.1%	98.1%	98.2%	99.9%	0.981
Resnet18	97.2%	97.4%	97.1%	99.8%	0.969
Xception	95.8%	96.1%	95.6%	99.7%	0.947
ShuffleNet	96.7%	96.6%	96.8%	99.7%	0.991
DarkNet-53	98.2%	98.4%	98.1%	99.8%	0.982



(**a**)



(**b**)

Figure 3. Sample confusion matrices for the best (DenseNet-201) and the worst (SqueezeNet) performing models using 50% of the data for training. (**a**) DenseNet-201; (**b**) SqueezeNet.

Further insight into the results is provided by Figure 4, which shows the mean, minimum, and maximum accuracy for all algorithms over 10 randomized runs for the 50/50 data split. Three models (i.e., SqueezeNet, GoogLeNet, and Darknet53) experienced high variability over the 10 random runs in comparison with the other models, which indicates their relative sensitivity to the choice of images included in the training/validations sets. The maximum standard deviation was 2.0% for SqueezeNet. The highest average accuracy was 98.8% for DesneNet-201.

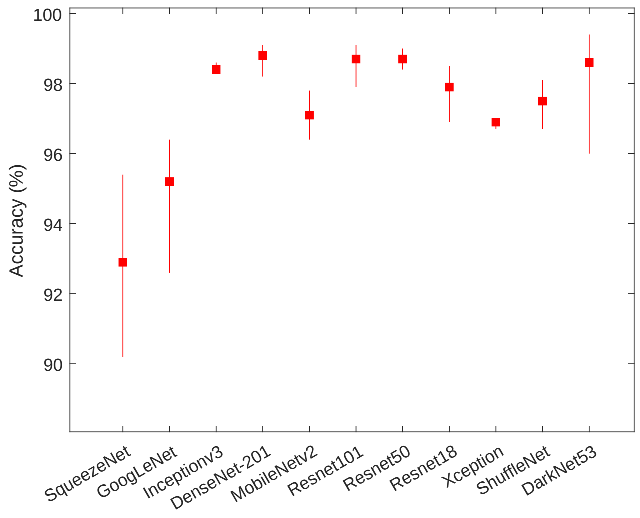
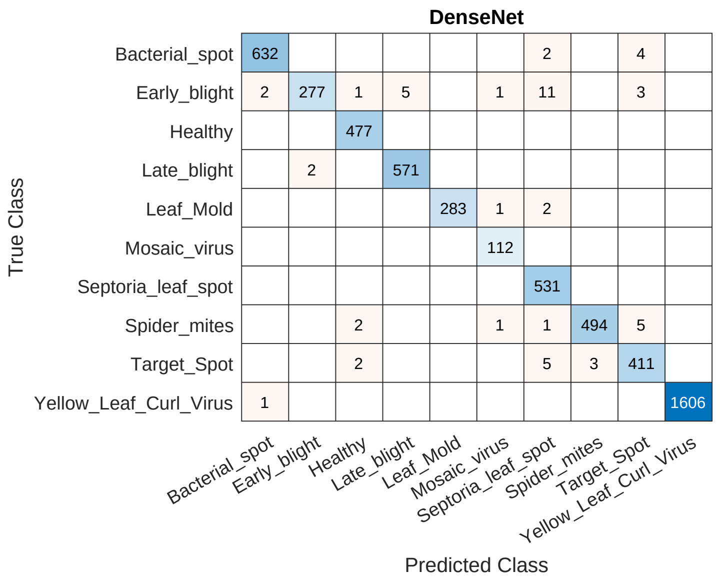


Figure 4. The mean, minimum, and maximum accuracy for all algorithms over 10 randomized runs and 50/50 data split.

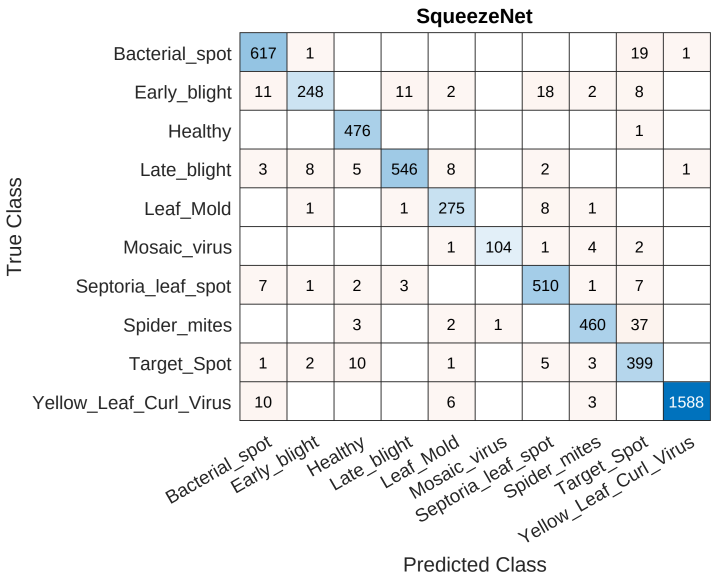
Although the number of images is somewhat acceptable considering the corresponding results, it is worthwhile to explore the effect of increasing the size of the training dataset. Deep learning models, in comparison to traditional machine learning algorithms, are well-known to achieve better performance with more data. Table 2 shows the mean over 10 runs for the overall F1 score, precision, recall, specificity, and MCC using 70% of the data for training. All models achieved better performance although with diminishing returns. SqueezeNet improved to 91.8% F1 score and DenseNet201 performed the best with an F1 score of 99.0%. The confusion matrices in Figure 5 corroborate the performance values and reveal a drastically improved diagnosis in comparison with the matrix in Figure 3 with relation to misclassifying spider mites and target spots as healthy. Figure 6 shows the fluctuation of the accuracy results for the eleven models over 10 randomized runs. In comparison to Figure 4, Darknet-53 displayed much less fluctuation with more training data, which means the model had the potential for better learning with more data. Most of the other models experienced less fluctuation; however, the smaller models (i.e., SqueezeNet and GoogleNet) do not seem to benefit from more training data with respect to their sensitivity to the random choices of the images to be included in the training data. The standard deviation of the accuracy results remained 2.0% for SqueezeNet. The highest average accuracy was 99.2% for DenseNet-201 and Darknet-53.

Table 2. The mean overall F1 score, Precision, Recall, Specificity, and MCC using 70% of the data for training. The results are an average of 10 runs.

Model	F1 Score	Precision	Recall	Specificity	MCC
SqueezeNet	91.8%	92.2%	92.2%	99.3%	0.914
GoogLeNet	94.8%	95.2%	94.8%	99.6%	0.937
Inceptionv3	98.5%	98.6%	98.4%	99.9%	0.978
DenseNet-201	99.0%	99.1%	98.9%	99.9%	0.99
MobileNetv2	97.1%	97.3%	97.1%	99.8%	0.957
Resnet101	98.9%	98.9%	98.9%	99.9%	0.987
Resnet50	98.6%	98.7%	98.6%	99.9%	0.98
Resnet18	97.8%	97.9%	97.8%	99.8%	0.976
Xception	96.7%	97.0%	96.5%	99.7%	0.965
ShuffleNet	97.4%	97.6%	97.3%	99.8%	0.973
DarkNet-53	98.9%	99.0%	98.9%	99.9%	0.987



(**a**)



(**b**)

Figure 5. Sample confusion matrices for the best (DenseNet-201) and the worst (SqueezeNet) performing models using 70% of the data for training. (**a**) DenseNet-201; (**b**) SqueezeNet.

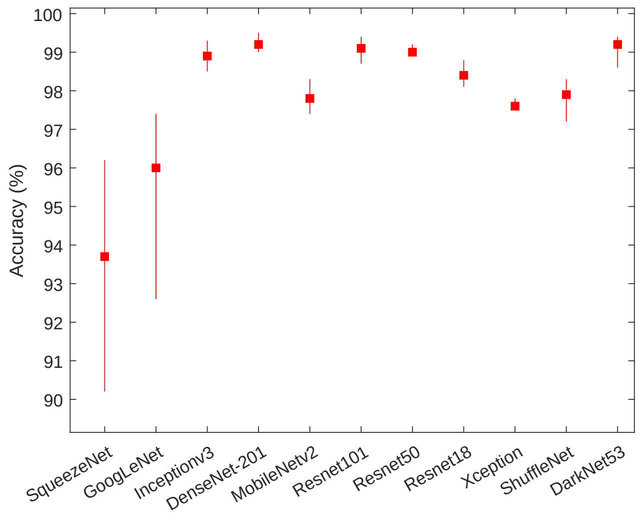


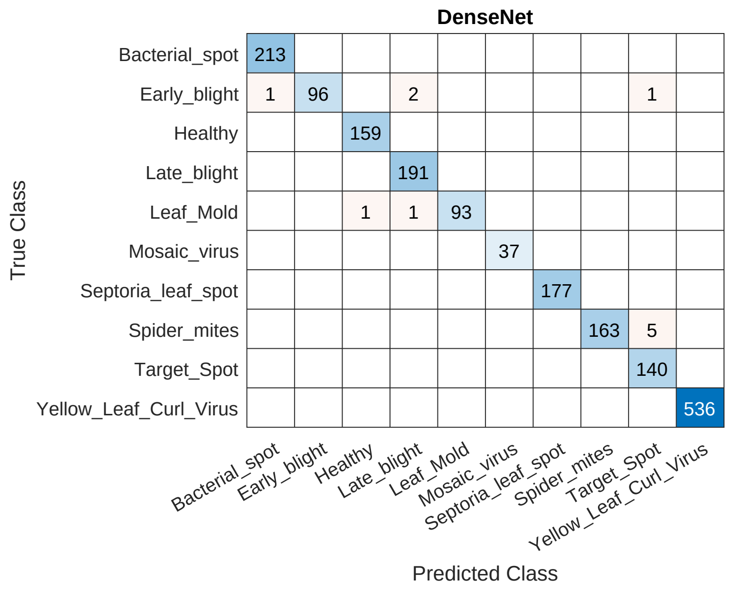
Figure 6. The mean, minimum, and maximum accuracy for all algorithms over 10 randomized runs and 70/30 data split.

Pushing toward the extreme case of using 90% of the images for training reveals further insight into the models. Table 3 shows the mean over 10 runs for the F1 score, precision, recall, specificity, and MCC using 90% of the data for training. Both SqueezeNet and GoogLeNet improved further to an F1 score of 93.3% and 95.9%, respectively. However, the other models with high performance values seemed to peek. Darknet53 did not improve and the remaining algorithms showed small improvements (i.e., <1%). DenseNet-201 achieved the maximum mean F1 score of 99.2% and was closely followed by Inceptionv3 at 99.1%. Figure 7 shows sample confusion matrices for the DensNet-201 and SqueezeNet models using 90% of the data for training. The figure shows that very few images were misclassified. DenseNet-201 classified several categories perfectly. Another observation relates to the ResNet models (101, 50, and 18) with larger numbers in the model's name corresponding to a deeper network; the models' performance improved with an increased depth and number of layers.

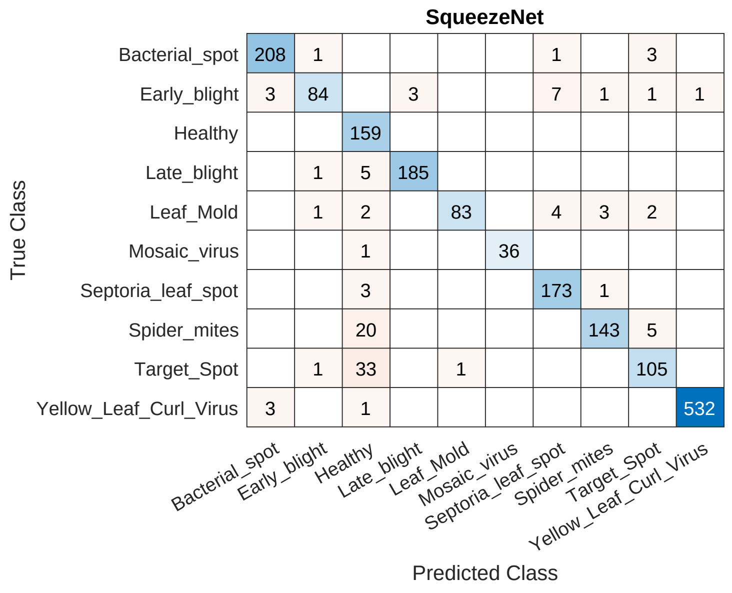
Table 3. The mean overall F1 score, Precision, Recall, Specificity, and MCC using 90% of the data for training. The results are an average of 10 runs.

Model	F1 Score	Precision	Recall	Specificity	MCC
SqueezeNet	93.3%	93.8%	93.3%	99.4%	0.93
GoogLeNet	95.9%	96.2%	95.8%	99.7%	0.942
Inceptionv3	99.1%	99.2%	99.0%	99.9%	0.99
DenseNet-201	99.2%	99.3%	99.1%	99.9%	0.991
MobileNetv2	97.8%	98.0%	97.8%	99.8%	0.975
Resnet101	99.0%	98.9%	99.1%	99.9%	0.99
Resnet50	98.8%	99.0%	98.7%	99.9%	0.986
Resnet18	98.2%	98.4%	98.2%	99.9%	0.981
Xception	97.1%	97.7%	96.7%	99.8%	0.971
ShuffleNet	97.8%	98.3%	97.6%	99.8%	0.975
DarkNet-53	98.9%	99.0%	98.8%	99.9%	0.986

Regarding the fluctuation of the results with different random choices, Figure 8 shows that SqueezeNet improved to 0.4% standard deviation for the classification accuracy, but GoogleNet had the highest standard deviation with 1.0%. The ShuffleNet model fluctuation does not seem to be affected by more training data and remained almost fixed throughout the various data splitting strategies. The highest average classification accuracy was 99.4% using DensNet-201.



(**a**)



(**b**)

Figure 7. Sample confusion matrices for the DensNet-201 and SqueezeNet models using 90% of the data for training. (**a**) DenseNet-201; (**b**) SqueezeNet.

Table 4 shows the mean training and validation times for all the models using 50/50, 70/30, and 90/10 data split. The SqueezeNet model trains the fastest in comparison to all other models. However, it also performs the worst. On the other hand, the Resnet18 seems to represent a good compromise between better classification performance and faster training time. The model produced a range of F1 scores of 97.2–98.2% with a corresponding training time of 395.5–491.9 s, which is very fast in comparison to the better performing models. Nonetheless, training times may not affect the ability to deploy the models in real-life applications, especially if no live model update is performed. This is because testing does not involve model update and is usually very fast, and training is done once and offline with respect to the deployment. The inference times were in the range of 0.5–7 millisecond/image, which is very small from a human user perspective. These times are independent of the data split and depend on the hardware planform and size of the model.

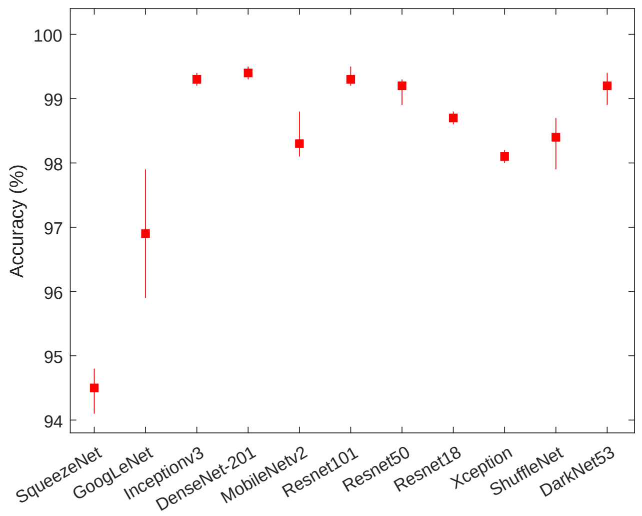


Figure 8. The mean, minimum, and maximum accuracy for all algorithms over 10 randomized runs and 90/10 data split.

Table 4. The mean training and validation times for all algorithms and data split strategies. All times are in seconds.

Data Split / **Model**	**50/50**	**70/30**	**90/10**
SqueezeNet	353.7	398.5	422.87
GoogLeNet	614.5	736.2	824.9
Inceptionv3	1796.5	2241.1	2595.3
DenseNet-201	5852.9	7311.8	8329.7
MobileNetv2	2455.9	3133.5	3691.3
Resnet101	5511.3	6951.9	8126.9
Resnet50	851.1	1054.9	1210.9
Resnet18	395.5	452.0	491.9
Xception	6582.1	8380.0	10,009.1
ShuffleNet	1822.8	2239.0	2717.8
DarkNet-53	1449.44	1722.0	2038.3

Several studies were conducted in the literature on the application of machine learning and deep learning algorithms for the identification and classification of plant diseases. Some of these studies (e.g., Hlaing and Zaw [11] and Annabel and Muthulakshmi [14]) used the traditional approach of employing image processing techniques to segment the input images (i.e., separation of the leaf or infected area from the background) and to extract texture features that reflect the disease state of the leaf. These features form the input for non-deep traditional machine learning algorithms (e.g., SVM). However, these studies did not consider images of different backgrounds and the classification performance results were worse than their deep learning counterparts. On the other hand, deep learning algorithms do not require these preprocessing steps and the accompanying overhead and errors. Agarwal et al. [15] modified the well-established structure of the VGG16 model and produced good performance. However, the original VGG16 model has shown its worth over hundreds of applications and thousands of studies and any modification will need to go through rigorous scrutiny. Tm et al. [13] used a similar approach to ours; however, the comparison was performed for three weaker models only (i.e., AlexNet, GoogleNet, and LeNet). Similarly, Kumar and Vani [12] experimente with four models (i.e., LeNet, VGG16, ResNet, and Xception) and produced 99.25% accuracy. However, their results were based on 14,903 leaf images from the same dataset with no apparent reason for dropping the remaining 3257 images. Table 5 shows a summary of the related literature to identify and classify tomato disease.

Table 5. A summary of the related literature to identify and classify tomato diseases. The size of the training subset is a percentage of the dataset.

Study	Aim	Data Source	No. of Images	Training Subset	Method	Performance
Mim et al. [10]	Classification (five disease + one healthy).	PlantVillage [23]	6000 tomato leaf images.	80%.	CNN.	Accuracy = 96.55%.
Hlaing and Zaw [11]	Classification (six disease + one healthy).	PlantVillage [23].	3474 tomato leaf images.	90/10 cross validation.	Feature extraction + SVM.	Accuracy = 84.7%.
Kumar and Vani [12]	Classification (nine diseases + one healthy).	PlantVillage [23].	14,903 tomato leaf images.	90%.	LeNet, VGG16, ResNet and Xception.	Accuracy = 99.25%.
Tm et al. [13]	Classification (nine diseases + one healthy).	PlantVillage [23].	18,160 leaf images.	13,360 images.	AlexNet, GoogleNet, and LeNet.	Accuracy = 94–95%.
Annabel and Muthulakshmi [14]	Classification (three diseases + one healthy).	PlantVillage [23].	2175 tomato leaf images.	70%.	Segmentation + feature extraction + random forest.	Accuracy = 94.1%.
Agarwal et al. [15]	Classification (nine diseases + one healthy).	PlantVillage [23].	18,160 leaf images.	1400 images per class.	Modified VGG16 structure.	Accuracy = 98.4%.
Liu and Wang [19]	Detection of gray leaf spot in leaf images.	Locally collected.	2385 plant images.	80%.	YOLOv3.	F1 score = 92.72%
Alhaj Ali et al. [17]	Classification (nine diseases + one healthy).	PlantVillage [23].	Classification (nine diseases + one healthy).	Inceptionv3.	Highest accuracy = 99.8%.	
Wang et al. [20]	Identification of disease type and infected area.	Collected from Internet.	286 fruit images.	60%.	Faster R-CNN and Mast R-CNN.	Mean average precision = 99.6%.
Lu et al. [22]	Classification (three diseases + one healthy).	Locally collected.	445 leaf images.	Spectral vegetation indices and KNN.	Highest accuracy = 100%.	
Gadade and Kirange [21]	Classification (six diseases + one healthy).	PlantVillage [23].	500 leaf images.	-	Feature extraction + SVM,KNN, NB, and DT.	Accuracy= 74%.
Al-gaashani et al. [18]	Classification (five diseases + one healthy).	PlantVillage [23].	1152 leaf images.	75%.	Feature extraction using MobileNetv2 and NASNetMobile + multinomial logistic regression.	Accuracy = 97%.
Ouhami et al. [16]	Classification (six diseases).	Locally collected.	666 leaf images.	80%.	DensNet161, DenseNet121, and VGG16.	Highest accuracy = 95.65% (DenseNet161).
This work	Classification (nine diseases + one healthy).	PlantVillage [23].	18,160 leaf images.	50%, 70%, and 90%.	Transfer learning using eleven CNN models.	Accuracy = 99.4% using 90% for training.

The present study has some limitations. First, tomato has two major leaf shapes (regular and potato leaf) and multiple other variations relating to leaf dimensions, color, and shades of green. However, the dataset does not include varieties of tomato leaf shapes. This will narrow the applicability and performance of any tomato disease identification system to the specific tomato variant in the dataset. Second, all the images in the dataset have a unified background. It would be worthwhile to investigate leaf images with different backgrounds taken in a non-unified manner. Third, tomatoes are susceptible to other diseases or pests (e.g., Tuta absoluta) that are not part of the dataset. Fourth, the dataset is imbalanced with varying numbers of images in each class.

4. Conclusions

Tomato is an important mass-produced agricultural product that is susceptible to diseases and the consequent yield loss. The use of deep transfer learning and well-established models showed a great potential in many applications in the literature. In this work, we targeted the identification of tomato diseases from infected leaf images. Using leaf images as input, eleven deep learning models were customized and retrained to identify nine tomato diseases in addition to healthy plants. The models (i.e., DarkNet-53, DenseNet-201, GoogLeNet, Inceptionv3, MobileNetv2,ResNet-18, ResNet-50, and ResNet-101, ShuffleNet, SqueezeNet, and Xception) were compared in terms of six common metrics and training/validation times. Although all models performed well, the DenseNet-201 model produced the best results with values larger than 99% for all metrics. However, the SqueezeNet model trained the fastest, and had the shortest inference time (i.e., 0.50 milliseconds/image).

The transfer learning approach carries inherent credibility and less complexity. In addition, it does not require explicit image processing nor feature extraction. Thus, it is suitable to be implemented in standalone smartphone applications, which can aid plant pathologists and farmers in quick and effective disease recognition and control. Future work will consider evolving the models by using incremental learning (i.e., improving the model during deployment). Moreover, the same approach can be adapted to identify diseases from tomato fruit images rather than the leaves. This may require 3D deep learning models to cover all sides of the image. In addition, other models or an ensemble of models can be used for solving the same problem. Field testing and commercial availability in the form of ready-to-download applications are promising areas of future activities.

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

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Support for the article processing fees was provided by Jordan University of Science and Technology, Deanship of Research project number 20210047.

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

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial intelligence
CNN	Convolutional neural networks
DT	Decision trees
NB	Naive Bayes
KNN	K-nearest neighbors
SVM	Support vector machine
FAO	Food and agriculture organization
T_P	True positive
T_N	True negative
F_N	False negative
F_P	False positive
ReLU	Rectified linear unit
TPR	True positive rate
N	Negatives
P	Positives
MCC	Matthews Correlation Coefficient
SGDM	Stochastic gradient descent with momentum
YOLOv3	employed you only look once version 3

References

1. Thangaraj, R.; Anandamurugan, S.; Pandiyan, P.; Kaliappan, V.K. Artificial intelligence in tomato leaf disease detection: A comprehensive review and discussion. *J. Plant Dis. Prot.* **2021**, *129*, 469–488. [CrossRef]
2. Vadivel, T.; Suguna, R. Automatic recognition of tomato leaf disease using fast enhanced learning with image processing. *Acta Agric. Scand. Sect. B Soil Plant Sci.* **2021**, *72*, 312–324. [CrossRef]
3. Durmuş, H.; Güneş, E.O.; Kırcı, M. Disease detection on the leaves of the tomato plants by using deep learning. In Proceedings of the 2017 6th International Conference on Agro-Geoinformatics, Fairfax, VR, USA, 7–10 August 2017; pp. 1–5. [CrossRef]
4. Ma, J.; Li, X.; Wen, H.; Fu, Z.; Zhang, L. A key frame extraction method for processing greenhouse vegetables production monitoring video. *Comput. Electron. Agric.* **2015**, *111*, 92–102. [CrossRef]
5. Verma, S.; Chug, A.; Singh, A.P. Prediction Models for Identification and Diagnosis of Tomato Plant Diseases. In Proceedings of the 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Bangalore, India, 19–22 September 2018. [CrossRef]
6. Fraiwan, M.; Audat, Z.; Fraiwan, L.; Manasreh, T. Using deep transfer learning to detect scoliosis and spondylolisthesis from x-ray images. *PLoS ONE* **2022**, *17*, e0267851. [CrossRef] [PubMed]
7. Khasawneh, N.; Fraiwan, M.; Fraiwan, L.; Khassawneh, B.; Ibnian, A. Detection of COVID-19 from Chest X-ray Images Using Deep Convolutional Neural Networks. *Sensors* **2021**, *21*, 5940. [CrossRef] [PubMed]
8. Sharma, N.; Jain, V.; Mishra, A. An Analysis of Convolutional Neural Networks for Image Classification. *Procedia Comput. Sci.* **2018**, *132*, 377–384. [CrossRef]
9. Kim, H.E.; Cosa-Linan, A.; Santhanam, N.; Jannesari, M.; Maros, M.E.; Ganslandt, T. Transfer learning for medical image classification: A literature review. *BMC Med. Imaging* **2022**, *22*, 69. [CrossRef] [PubMed]
10. Mim, T.T.; Sheikh, M.H.; Shampa, R.A.; Reza, M.S.; Islam, M.S. Leaves Diseases Detection of Tomato Using Image Processing. In Proceedings of the 2019 8th International Conference System Modeling and Advancement in Research Trends (SMART), Moradabad, India, 22–23 November 2019. [CrossRef]
11. Hlaing, C.S.; Zaw, S.M.M. Model-Based Statistical Features for Mobile Phone Image of Tomato Plant Disease Classification. In Proceedings of the 2017 18th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), Taipei, Taiwan, 18–20 December 2017. [CrossRef]
12. Kumar, A.; Vani, M. Image Based Tomato Leaf Disease Detection. In Proceedings of the 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kanpur, India, 6–8 July 2019. [CrossRef]
13. Tm, P.; Pranathi, A.; SaiAshritha, K.; Chittaragi, N.B.; Koolagudi, S.G. Tomato Leaf Disease Detection Using Convolutional Neural Networks. In Proceedings of the 2018 Eleventh International Conference on Contemporary Computing (IC3), Noida, India, 2–4 August 2018. [CrossRef]
14. Annabel, L.S.P.; Muthulakshmi, V. AI-Powered Image-Based Tomato Leaf Disease Detection. In Proceedings of the 2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 12–14 December 2019. [CrossRef]
15. Agarwal, M.; Gupta, S.K.; Biswas, K. Development of Efficient CNN model for Tomato crop disease identification. *Sustain. Comput. Inform. Syst.* **2020**, *28*, 100407. [CrossRef]

16. Ouhami, M.; Es-Saady, Y.; Hajji, M.E.; Hafiane, A.; Canals, R.; Yassa, M.E. Deep Transfer Learning Models for Tomato Disease Detection. In *Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 65–73. [CrossRef]
17. Ali, A.A.; Chramcov, B.; Jasek, R.; Katta, R.; Krayem, S.; Awwama, E. Tomato Leaf Diseases Detection Using Deep Learning. In *Lecture Notes in Networks and Systems*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 199–208. [CrossRef]
18. Al-gaashani, M.S.A.M.; Shang, F.; Muthanna, M.S.A.; Khayyat, M.; El-Latif, A.A.A. Tomato leaf disease classification by exploiting transfer learning and feature concatenation. *IET Image Process.* **2022**, *16*, 913–925. [CrossRef]
19. Liu, J.; Wang, X. Early recognition of tomato gray leaf spot disease based on MobileNetv2-YOLOv3 model. *Plant Methods* **2020**, *16*, 83. [CrossRef] [PubMed]
20. Wang, Q.; Qi, F.; Sun, M.; Qu, J.; Xue, J. Identification of Tomato Disease Types and Detection of Infected Areas Based on Deep Convolutional Neural Networks and Object Detection Techniques. *Comput. Intell. Neurosci.* **2019**, *2019*, 9142753. [CrossRef] [PubMed]
21. Gadade, H.D.; Kirange, D.D. Machine Learning Approach towards Tomato Leaf Disease Classification. *Int. J. Adv. Trends Comput. Sci. Eng.* **2020**, *9*, 490–495. [CrossRef]
22. Lu, J.; Ehsani, R.; Shi, Y.; de Castro, A.I.; Wang, S. Detection of multi-tomato leaf diseases (late blight, target and bacterial spots) in different stages by using a spectral-based sensor. *Sci. Rep.* **2018**, *8*, 2793. [CrossRef] [PubMed]
23. Pandian, J.A.; Geetharamani, G. Data for: Identification of Plant Leaf Diseases Using a 9-layer Deep Convolutional Neural Network. *Mendeley Data* **2019**, *1*, 2019. [CrossRef]
24. Redmon, J. Darknet: Open Source Neural Networks in C, 2013–2016. Available online: https://pjreddie.com/darknet/ (accessed on 22 August 2022)
25. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2261–2269. [CrossRef]
26. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 1–9. [CrossRef]
27. Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A.A. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; AAAI Press: Palo Alto, CA, USA, 2017; pp. 4278–4284.
28. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
29. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6848–6856. [CrossRef]
30. Iandola, F.N.; Moskewicz, M.W.; Ashraf, K.; Han, S.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1 MB model size. *arXiv* **2016**, arXiv:1602.07360.
31. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 1800–1807. [CrossRef]
32. Qian, N. On the momentum term in gradient descent learning algorithms. *Neural Netw.* **1999**, *12*, 145–151. [CrossRef]
33. Shorten, C.; Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. *J. Big Data* **2019**, *6*, 60. [CrossRef]
34. Tharwat, A. Classification assessment methods. *Appl. Comput. Inform.* **2020**, *17*, 168–192. [CrossRef]
35. Gorodkin, J. Comparing two K-category assignments by a K-category correlation coefficient. *Comput. Biol. Chem.* **2004**, *28*, 367–374. [CrossRef] [PubMed]