Lecture Notes in Networks and Systems 288

Mukesh Saraswat
Sarbani Roy
Chandreyee Chowdhury
Amir H. Gandomi *Editors*

Proceedings of International Conference on Data Science and Applications

ICDSA 2021, Volume 1

Springer

Lecture Notes in Networks and Systems

Volume 288

Vigorous Deep Learning Models for Identifying Tomato Leaf Diseases

Rajeev Karothia and Manju K. Chattopadhyay MIEEE

Abstract Identifying and detecting a plant disease is a primary challenge in the farming segment. Specifically, for tomato farming: Early Blight, Late Blight, Mosaic Virus, Target Spot, Yellow Leaf Curl Virus (YLCV), Bacterial Spot, Leaf mold, Septoria leaf spot and two-spotted spider mite are nine general diseases that severely affect the yield. In this work, we propose convolutional neural networks (CNNs) based deep learning (DL) model for the identification of tomato leaf diseases. In our study, we take the Tomato Leaf Disease Dataset (TLDD) consisting of 18,160 images of the infected and the healthy tomato leaves from PlantVillage. We first select the most suitable and accurate deep learning models for disease identification experiments. Four popular models viz. SqueezeNet, ResNet50, InceptionV3 and DenseNet are chosen for the analysis. Lastly, using ImageAI library on Google Colaboratory (Colab), we train all the four models on the collected tomato-leaf-image dataset to identify the presence of above mentioned nine common tomato leaf diseases. Results from our experiments on the comparative study of the selected deep learning models identify nine different tomato leaf diseases. We infer that the InceptionV3 model provides the highest accuracy of 99.64%. Our chosen model provides faster detection with higher accuracy as compared to the other models. In future work, we intend to modify the algorithm to develop our own model for disease identification for other crops as well.

Keywords Deep learning · Convolutional neural network · Tomato leaf disease · ImageAI

1 Introduction

Many natural parameters affect production quality and quantity of crop yield viz. moisture, humidity, temperature, soil, etc. Crop diseases associated with these parameters play a major hindrance in good crop yield. Consequently, disease management

R. Karothia (✉) · M. K. Chattopadhyay MIEEE
School of Electronics, Devi Ahilya University, Madhya Pradesh, Indore 452017, India

M. Saraswat et al. (eds.), *Proceedings of International Conference on Data Science and Applications*, Lecture Notes in Networks and Systems 288,
https://doi.org/10.1007/978-981-16-5120-5_11

is one of the key issues in agro-economics to be taken care of. Diseases need to be detected at the initial stage itself to control their further spread with the help of proper treatment. With the advances in the technology, it is now possible to detect specific diseases by utilizing and monitoring the images of the diseased leaves of infected crop. Traditionally, plant diseases have been detected by experts by visual observation. This method besides being sluggish in nature, involves risk of misinterpretation. To expedite the process and make it more error-free, a variety of spectroscopic and imaging methods has been developed for plant leaf diseases recognition [1]. However, many such techniques deploy accurate instruments and precise sensors which make the system expensive and, often, infeasible.

Machine learning (ML) and deep learning are demonstrating their significance and usefulness in vast fields ranging from IT division to agriculture sector to web administrations and many other application areas. ML and DL are recent popular methods for image processing and data analysis that have gained strong foothold in the agriculture-domain for leaf infection detection [1–3]. Traditional ML algorithms have limitations in processing natural data in raw form. CNN based DL algorithms incorporate computational models that significantly enhance the cutting edge in visual, audio, video recognition, object and images processing [4–6]. CNNs algorithms are excellent choice for feature-extraction from the input images, with less complex preprocessing of images. Hence, these methods have set a research trend in image recognition and identification [7, 8]. They are also implemented for disease detection of crops such as potato, rice, mango, apple, maize, cassava, cucumber and tomato [9–19]. Nonetheless, the tomato leaf disease detection is still stubborn because of its peculiar characteristics: Firstly, the size of the visual spots on the infected leaves differ with the same as well as with the different diseases. Maximum numbers of spots are very tiny in size. Secondly, multiple diseases may occur on a single leaf. Lastly, ecological factors such as air temperature, air humidity, UV index, solar radiation and soil health also hamper the disease detection.

To deal with these challenges, we compare and train DL algorithms based on CNN, to investigate and identify leaf diseases in tomato. The major roles of this paper are summarized as follows:

- We have collected healthy and diseased tomato leaf dataset from PlantVillage by Kaggle [20]. This provides promise of generalization potential of our proposed model. As the numbers of available diseased tomato leaf images is insufficient, we perform data augmentation to overcome the overfitting problem in the model training process.
- An application is developed to train our disease detection model based upon Python library ImageAI [21]. It has classes and methods to train image dataset and to generate a new model that can be used to perform prediction or detection on custom leaf objects. First, we use ModelTraining() class to train our dataset on DL algorithms viz. SqueezeNet, ResNet50, InceptionV3 and DenseNet [21]. A JSON file is generated after the model training process. This file contains detail of all the object-types contained in the dataset. Based upon the training output

and results, the model with the maximum accuracy is chosen. Using the chosen model, we do image-identification and then, generate the JSON file.

- All of our selected models are based on CNN. Our proposed new custom model identifies the different features of the diseased tomato leaf and detects the nine types of tomato leaf diseases with best accuracy. In addition, our proposed approach can handle multiple images for disease detection.
- Our experiments show that the different models have different accuracies. For chosen dataset, we have achieved highest 99.64% accuracy with InceptionV3 model and lowest 99.41% accuracy in ResNet50 model. Our proposed all new models also demonstrate strong recognition results.

Remaining part of this paper has been structured in the following manner: In Sect. 2, literature survey and related works are summarized. Details about our materials and methods are given in Sect. 3. Section 4 describes different detection models for tomato leaf diseases in detail. In Sect. 5, analysis and discussion of experimental results for performance evolution of our approach are provided. Finally, the paper concludes with Sect. 6.

2 Literature Review

Mohanty et al. used leaf dataset of 54, 306 photographs to train a convolutional neural network which recognized 26 diseases and 14 crop species [9]. Their model achieved a precision of 99.35%. They measured the performance of trained models based on correct crop disease-pair detection. Lu et al. described a new CNNs technique to identify rice diseases [10]. They used total 500 natural images in dataset of healthy and infected leaves of rice. This model was trained to detect 10 common rice diseases. To train their CNNs model, they have used back gradient-descent algorithm with an accuracy of 95.48% which is much superior to traditional machine learning model. Singh et al. presented an innovative model named multilayer convolutional neural network to detect fungal disease Anthracnose for the classification of Mango [11]. They captured real-time dataset which consist of 1,070 images. There model achieved performance with accuracy of 97.13%. In Ref. [12], Barbedo et al. studied deep learning in the plant pathology. Their experiment was performed on an open source dataset which contained 50,000 samples of 171 different diseases for 21 type's species. However, the authors selected only corn diseases images for their study. They explained extrinsic factors such as limited annotated dataset, image background, image capture conditions and covariate shift with some intrinsic factors such as symptom variations, simultaneous disorders and symptom segmentation. These factors impact the performance of CNNs.

Jaing et al. used an apple leaf diseased dataset containing 26,377 images and proposed single shot detector (SSD) which includes inception and rainbow concatenation (INAR-SSD) model to train dataset [13]. Their results show that this model recognizes a detection accuracy of 78.80% mAP on apple leaf disease dataset. In

Ref. [14], authors worked on maize leaf diseases detection. They proposed CNN based two improved models, viz. GoogLeNet and Cifar10, for nine different type of diseases in maize leaves. The disease detection accuracy of the GoogLeNet model and Cifar10 model was 98.9% and 98.8%, respectively. In Ref. [15], Ferentinos et al. trained AlexNet, AlexNetOWTBn, GoogLeNet, Overfeat and VGG CNN architectures using database of 87, 848 samples consisting of 25 diverse plants that contained 58 combinations of plants and diseases. They trained and tested all models on torch ML scientific computing framework. Their experimental results demonstrated that the VGG model has a recognition rate of 99.53%.

Ramcharan et al. have used transfer learning methods to trainInceptionV3 model to detect 3 different types of cassava leaf diseases [16]. They trained this model on 11,670 cassava images and found overall accuracy of 93%. Ma et al. have used deep CNN (DCNN) techniques to recognize four major cucumber diseases (i.e., Anthracnose, Target leaf spots, Powdery mildew and Downy mildew) [17]. Author used cucumber leaf datasets containing 14,208 photographs, and trained model accuracy of DCNN is 93.4% on unbalanced data.

Single shot multibox detector (SSD), faster region-based and region-based fully CNN deep learning network's comparative study were done by Fuentes et al. to recognize tomato leaf diseases [18]. Their trial results show that this technique efficiently catches tomato disease and might handle background variations. Furthermore, they state that by using data annotation and augmentation, and they can achieve better performance. They have used only 5000 images of tomato leaf to perform this study. Kumar et al. [19] deployed SVM for plant disease identification with swarm-based approach.

All these varied studies show that the CNNs have been used broadly in the field of plant leaves disease identification and are producing acceptable results. Table 1 summarizes the various findings discussed above.

For our research work, we have chosen tomato plant for the studies, as it is viewed as one of the most beneficial crops in India. This crop can be produced and delivered throughout the year and its demand in domestic and worldwide markets keeps the business alive. After China, India is the second largest producer of tomatoes crop in the world. Total tomato production, worldwide, is around 1300 lakh tones [22]. Tomatoes are the most important protective food used for soup, salad, pickles, ketchup, puree, sauces and in many other ways because of its special nutritive value.

Therefore, in our work, we trained four different disease detection CNNs models using ImageAI python library on Google Colab to increase the tomato leaf disease detection accuracy. To achieve better performance and accuracy, we use total 18,160 images and four CNN models with data annotation and augmentation techniques. With this experiment, we achieve accuracy of 99.64% in InceptionV3 Model. After identifying the best model, we intend to modify it in our future work, for detection of disease in other crops.

Table 1 Summary of research work on plant/leaf disease recognition using CNNs techniques

Author	Network used	Plant name	Number of diseases detected	Dataset used	Accuracy achieved
Mohanty et al.	CNN	14 crop species	26	Public dataset	99.35%
Yang Lu et al.	CNN	Rice	10	Self	95.48%
U. P. Singh et al.	Multilayer CNN	Mango	single	Self	97.13%
Barbedo et al.	GoogLeNet	Multiple	Multiple	Digipathos	80.75%
Jiang et al.	INAR-SSD	Apple	5	Self and PlantVillage	78.80% mAP
Zhang et al.	GoogLeNet and Cifar10	Maize	Multiple	PlantVillage and Google	98.9% and 98.8%
Ferentinos et al.	CNN	Multiple	Multiple	PlantVillage	99.53%
Ramcharan et al.	CNNs Inception v3	cassava	3	PlantVillage	93%
Ma et al.	DCNN	cucumber	Multiple	Self	93.4%
Fuentes et al.	SSD, R-FCN and Faster R-CNN	Tomato	9	Self	85.98% mAP

3 Materials and Methods

3.1 *Overview*

For image recognition based research, an appropriate dataset is required throughout the model training for performance evolution of the algorithm [15]. The detailed disease detection flowchart is presented in Fig. 1. First, the TLDD is prepared by collecting infected and healthy photographs from PlantVillage [20]. TLDD is annotated and expanded via a sequence of data augmentation procedures. Then, the dataset is separated into two sections: (i) The training data, which is used to train the different CNNs models and (ii) the testing data, which is used for performance valuation. Then, we validate and compare the results of all our selected CNNs models with each other.

3.2 *Tomato Leaf Disease Dataset (TLDD)*

We take the images from PlantVillage Dataset [20] corresponding to following ten classes:

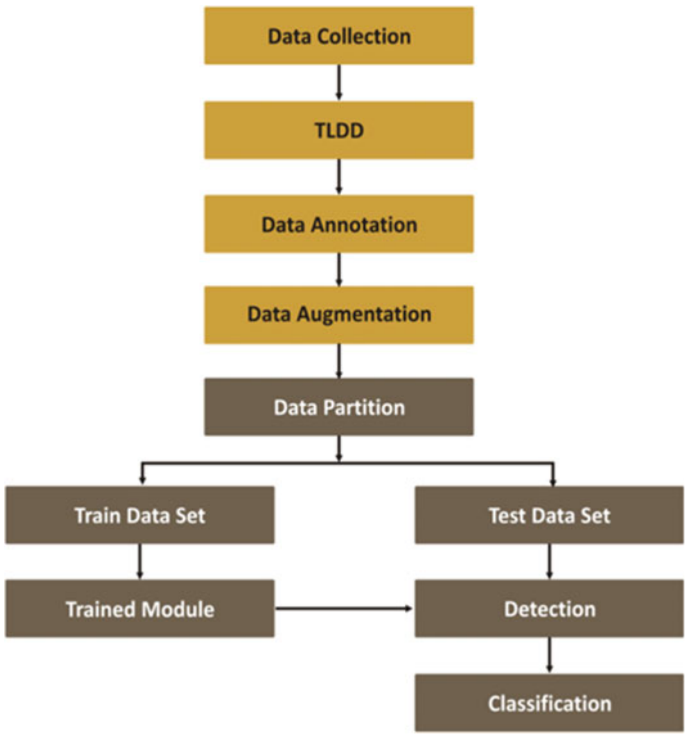


Fig. 1 Flow chart for leaf disease detection

1. Target spot
2. Mosaic virus
3. Yellow leaf curl virus (YLCV)
4. Bacterial spot
5. Early blight
6. Late blight
7. Leaf mold
8. Septoria leaf spot
9. Spider_mites two-spotted_spider_mite
10. Healthy leaves

Figure 2a–j depicts the diseased and healthy tomato leaves images. The nine general tomato leaf diseases are selected for following reasons: Foremost, these viruses can be visually recognized on leaves—an important aspect with respect to CNNs application. Besides, these diseases cause considerable crop diminution in the tomato industry. This dataset includes images of wide-ranging resolutions, at different stages of disease (like at early, medium, or severe infection), of diverse leaf sizes. And, images have different infected areas on the leaf.

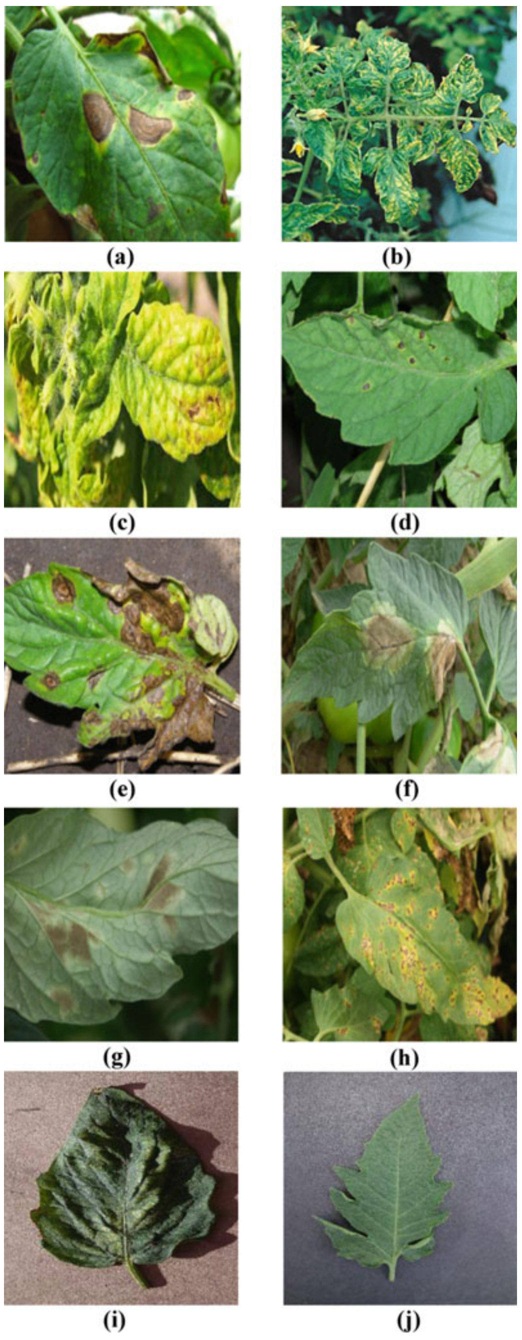


Fig. 2 Tomato leaves: **a–i** Diseased leaves, **j** healthy one (Emmanuel 2018)

3.3 Data Augmentation

This is a very important process in deep learning, especially, when the numbers of pictures in the data are inadequate. This method is generally used to enhance the image dataset. Data augmentation also helps to overcome the drawback of deep neural networks system, i.e., the overfitting problem. Overfitting frequently refers to the number of total images that are used for training dataset or for the hyper-parameters' selection [13]. Many techniques are used for data augmentation: image rotation, vertical and horizontal flips, changes in color intensity, changes in intensity, contrast and sharpness. As depicted in Fig. 3, with the help of data augmentation, we can generate new images from original images. This process helps to increase images in dataset and provide better results for training models. To perform data augmentation, we use the Keras library which provides the capability of image data augmentation using following class:

```
from keras.preprocessing.image import ImageDataGenerator
NewdataGen = ImageDataGenerator(
    rotation_range = 90, width_shift_range = 0.1,
    height_shift_range = 0.1, rescale = 1./255,
    shear_range = 0.1, zoom_range = 0.1,
    horizontal_flip = True, fill_mode = 'nearest').
```

4 Disease Detection Models for Tomato Leaf

Artificial intelligence has been frequently used to improve agricultural produce, storage and analytics since the rise of machine learning and deep learning [23]. This helps effectively to:

- Get accurate health of crops
- Detect plant diseases
- Automate harvesting and crop sorting
- Acquire real-time data on soil conditions
- Support the implementation of precision irrigation

Here, we explain how we can implement artificial intelligence using computer vision and deep learning to automate tomato leaf disease detection of damaged ones. In this study, we use imageAI python library which provides simple and easy way to use classes to train CNN based deep learning algorithms like **SqueezeNet, ResNet50, DenseNet and InceptionV3**, on user's datasets using only a small number of lines of program to generate custom models [21]. After we train our dataset using these libraries, we generate new models for image detection. Later, we can apply the "**CustomImagePrediction**" class to predict /identify a picture or a set of pictures. The "**ModelTraining**" class allows user to train the four different deep learning

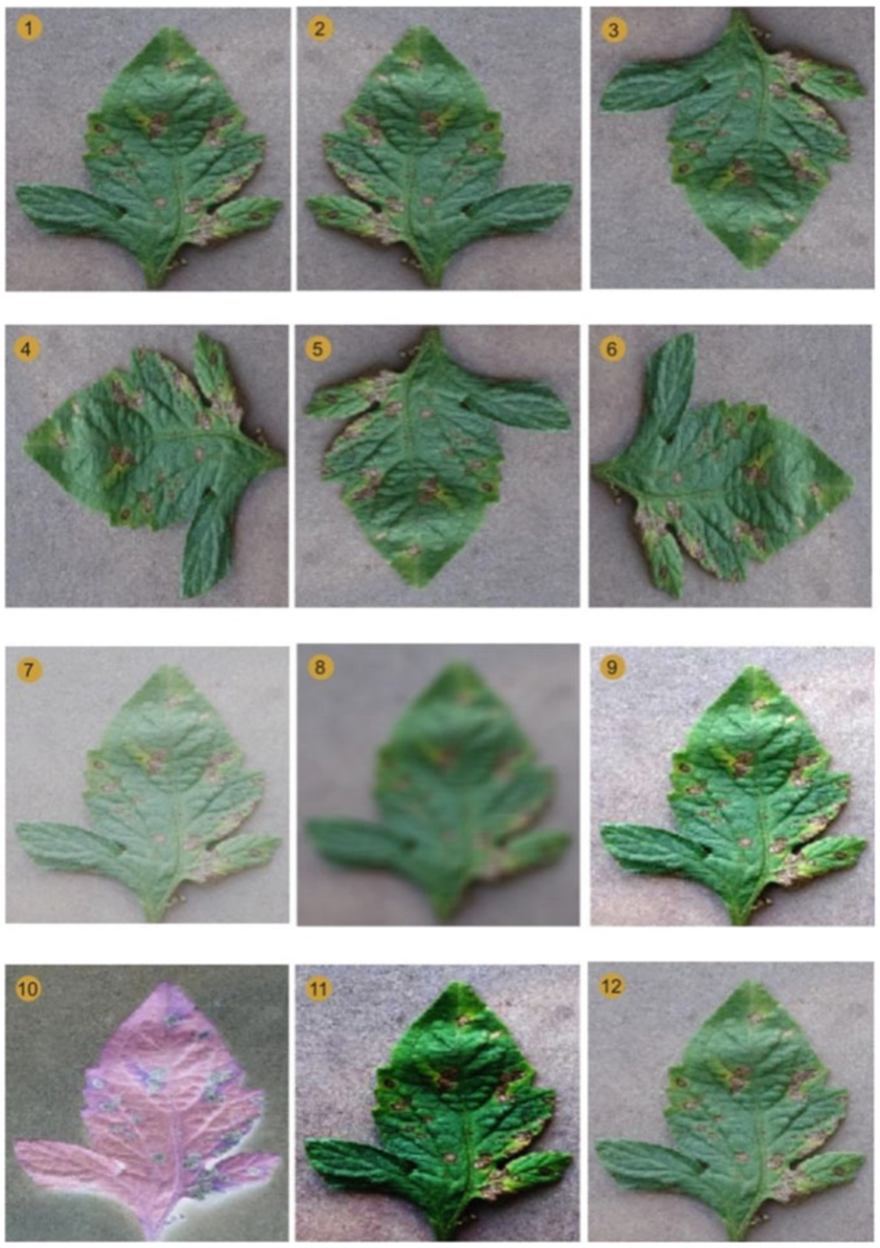


Fig. 3 Augmentation of tomato leaves: (1) simple image; (2) horizontal flip; (3) vertical flip; (4) 90° shift; (5) 180° shift; (6) 270° shift; (7) gamma adjustment; (8) blurred image; (9) high intensity; (10) image inversion; (11) sigma corrected image and; (12) logarithmic corrected image

algorithms (**ResNet50**, **InceptionV3**, **SqueezeNet** and **DenseNet**) on any image data to create new models for prediction.

4.1 SqueezNet Model

The SqueezeNet [24] is a tiny CNN architecture that requires less number of parameters to maintain the accuracy. This architecture has 26 convolutional layers (with 34 other types of layers) and about 860 million multiply-accumulate operations. This CNN architecture suggests following advantages:

- It needs less communication between servers throughout distributed training.
- This model needs less bandwidth to export a new model from the cloud to the client.
- It requires limited memory resources to get installed on hardware.

The building brick of SqueezeNet is also termed as fire module. This architecture has various expanded and squeezed layers [24].

Figure 4 depicts the detailed architecture of SqueezeNet. SqueezeNet (Left) that starts with a single-convolution layer (conv1), then eight fire modules (fire2 to fire9). Last layer is also the convolution layer (conv10). The number of filters after every fire module gradually increases from start to the end of the complete network. Also, we add and implement max-pool function with a stride of two, after conv1, fire4, fire8 and conv10 layer. SqueezeNet with simple bypass (Middle) and SqueezeNet with complex bypass (Right) are also shown in Fig. 4.

4.2 Residual Network (ResNet) Model

It is a convolutional neural network that builds from pyramidal cells in the cerebral cortex. ResNets breaks down a deep plain neural network into tiny chunks of network connected through skip or shortcut connections to make an even bigger network [23]. Generally, these models are constructed with two or three layer skips which contain nonlinearities (ReLU) and the batch normalization function in between weight layers.

Microsoft developed a deep residual learning framework to overcome degradation disadvantage in deep CNN. Rather than hoping for each stacked layer to directly fit into a desired underlying mapping, they explicitly let these layers fit a residual mapping. The theory of $F(x) + x$ is completed y feed-forward networks with "shortcut connections" shown in Fig. 5. In ResNet, shortcut skips one or more layers and these connections do not add extra parameter so as to avoid computational complexity. To understand the mathematical equation of the above mentioned ResNet block, we consider the X as the output of its previous (l)th layer in Eq. 1. Next, we calculate $F(X)$ for $(l + 1)$th layer (weighted layer 1) using Eq. 2. Now, Eq. 3 is the output of $(l + 1)$th layer where we use ReLu as an activation function. Equation 4 calculates

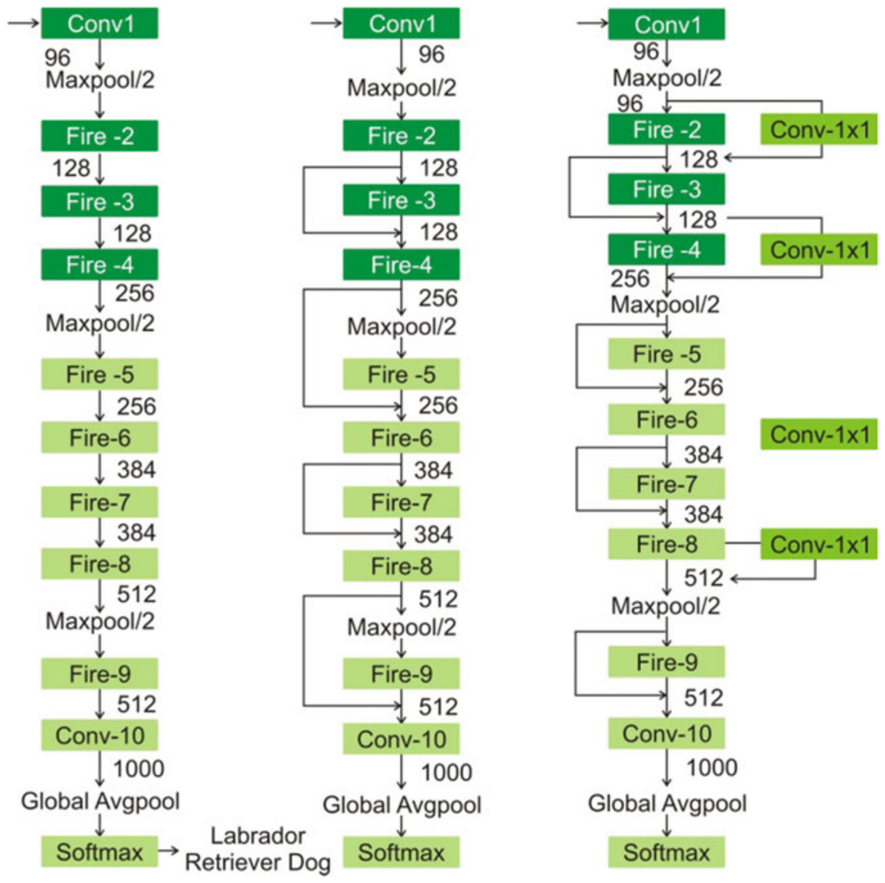
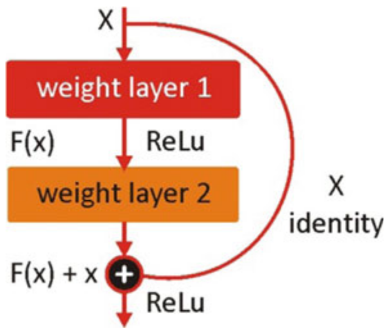


Fig. 4 SqueezeNet architecture

Fig. 5 ResNet architecture



$F(X)$ for $(l+2)$th layer (weight layer 2).

$$a^{[l]} = X \tag{1}$$

$$F(X) = Z^{[l+1]} = W^{[l+1]}.a^{[l]} + b^{[l]} \tag{2}$$

$$a^{[l+1]} = \text{ReLu}(Z^{[l+1]}) \tag{3}$$

$$F(X) = Z^{[l+2]} = W^{[l+2]}.a^{[l+1]} + b^{[l+2]} \tag{4}$$

$$a^{[l+2]} = \text{ReLu}(Z^{[l+2]} + a^{[l]}) \tag{5}$$

Finally, we take the ReLu of $F(X) + X$ in Eq. 5, as the output of this residual block.

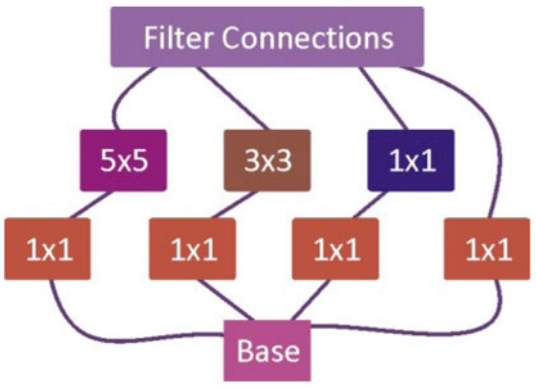
4.3 InceptionV3 Model

This micro-architecture was first initiated by Szegedy et al. in 2014 and was originally introduced as GoogLeNet, which is actually the Inception-v1. Afterward, the inception design was polished in different manners. First, modification involved an introduction of batch normalization by inserting extra factorization thoughts in the third cycle. This was called InceptionV3 [25]. In Fig. 6a, b, one can see two 3×3 convolutions replace the 5×5 convolution. This reduces the number of parameters for calculation. The purpose of this architecture is to reduce computational resources in DL based high-accuracy image classification.

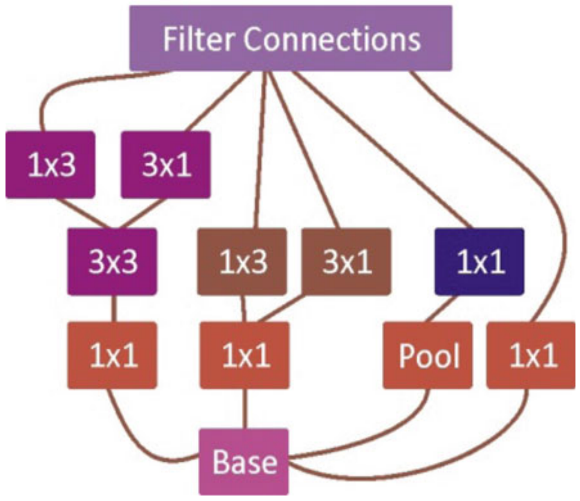
4.4 DenseNet Model

DenseNet (densely connected convolutional network) is the latest neural network for visual object recognition. It is quite similar to ResNet. However, in DenseNet, every layer takes additional inputs from all previous layers and transfers them to all the successive layers along with the feature-maps [26].

In Fig. 7, detailed architecture of DenseNet is shown with a five-layered dense block (X0–X4) with a growth rate of $k = 4$. Where k is the growth rate of additional features generated by each layer. Each layer takes all preceding feature-maps (H1–H4) as input. Every layer is receiving "collective information" from all former layers. In this type of architecture, network can be narrow and dense with fewer channels [27].



(a) Original InceptionV3 module structure



(b) InceptionV3 module with expanded filter banks

Fig. 6 Graphical structure ofInceptionV3 modules

5 Results and Discussions

In this section, we discuss about the hardware and software setup used for this study, followed by dataset description. In the later part, we analyze our experimental study.

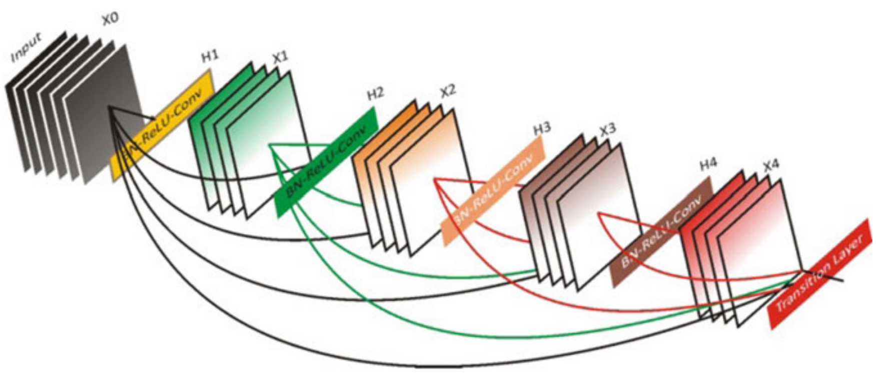


Fig. 7 Five layer DenseNet block architecture

5.1 Experimental Setup

For this experimental research, we used Windows Operating System with an Intel R Core(TM) i5-9300H CPU @ 2.40 GHz (8 CPUs) that was powered by an NVIDIA GTX1050 GPU. This GPU has 640 CUDA cores and 4 GB memory. The core frequency is up to 1354 MHz and the floating-point performance is 10.5 TFLOPS. We have used cloud-based Google Colab environment for python algorithm writing and execution. We use ImageAI which utilizes Tenser Flow as the backbone of computer vision operation along with the python libraries with Python 3.5.1, Tensorflow 1.4.0, OpenCV and Keras 2.x.

5.2 Dataset

To perform the experiment, we divide the dataset in 80: 20 ratios for training and testing, respectively. Detailed data of training and testing images of diseased and healthy leaves is given in Table 2. Our image dataset contains 10 different classes/types of images. To achieve highest accuracy, we must have minimum 500 images in each class. For the training process, we first create a folder for dataset. In that, we have to create two subfolders named *Test* and *Train*. In these two folders, we create subfolders of each object name, where images of object name are kept. Refer Fig. 8 for the arrangement of dataset folders. To perform all training and testing experiments, we use colaboratory (Colab) online cloud-based Jupyter notebooks environment which is developed by Google research project team to make work of researchers, students or data scientists easier. It allows user to execute python programs in any browser with zero configuration and provides online GPUs and TPUs to train our machine learning and deep learning algorithms and models.

Once dataset is prepared, we can initiate the model training process. We execute this training process on Google Colab with learning rate of 0.01, total 10 objects,

Table 2 Training and testing sets of the tomato leaves

Disease	Images		Total number
	Train	Test	
Target spot	1124	280	1404
Mosaic virus	299	74	373
YLCV	4286	1071	5357
Bacterial spot	1702	425	2127
Early blight	800	200	1000
Late blight	1528	381	1909
Leaf mold	762	190	952
Septoria leaf spot	1417	354	1771
Two-spotted_spider_mite	1341	335	1676
Healthy	1273	318	1591
Total	14,532	3628	18,160

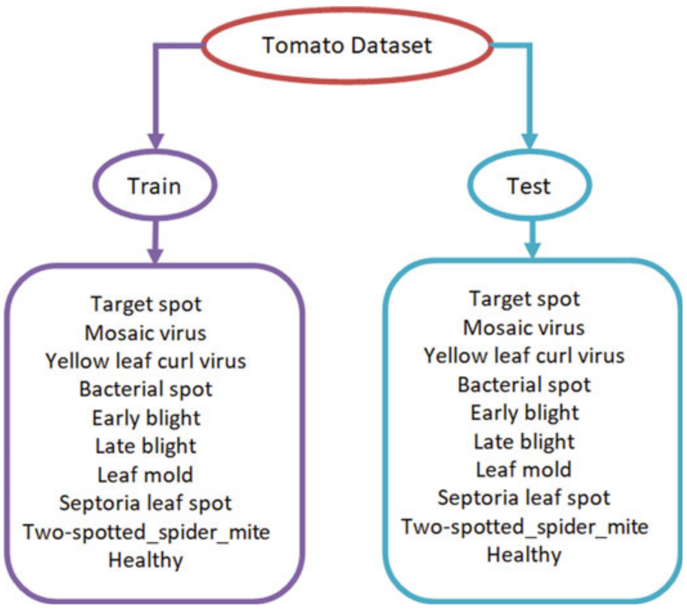


Fig. 8 Dataset folder arrangement

0.5 as dropout rate and total 25 Epochs. This process generates a JSON file which contains information of all the objects types in dataset and start creating new models. The process is continued till model with the highest accuracy is identified. Later, prediction process using the generated model is performed. Then, the performance of the proposed model is validated, and comparison of the outcomes of all trained models is done.

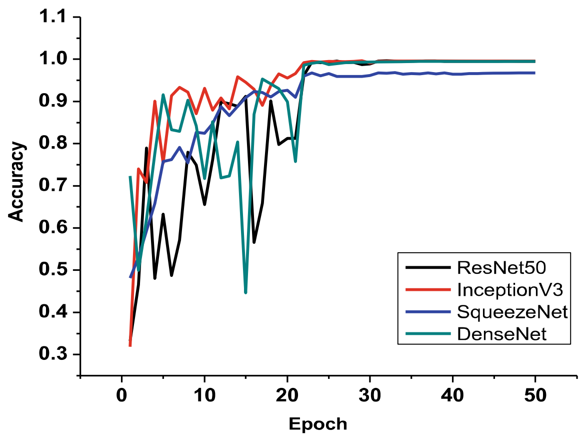


Fig. 9 Comparison of CNN model accuracy

Table 3 List of models recognition accuracy

S. No.	Model type	Accuracy
1	DenseNet	99.50
2	ResNet50	99.41
3	InceptionV3	99.64
4	SqueezeNet	96.85

5.3 Model's Accuracy

Here, we discuss model's image detection accuracy for SqueezeNet, ResNet and InceptionV3 and DenseNet deep convolution networks trained on TLDD. Training accuracy and training epoch graph are plotted in Fig. 9, where accuracy is on y-axis and iteration is on the x-axis. Please refer Table 3 for data related to maximum accuracy achieved by different models after training iteration. In our proposed approach, InceptionV3 model achieved the highest training model accuracy.

5.4 Training and Validation Accuracy and Loss

Once the model is built, it needs to be validated by inducing different datasets. Usually, constraints are faced in terms of amount of accurate data available for training. Validating the model is necessary so that reliability of the model can be evaluated through the validation dataset. We, therefore, evaluate trained model on validation dataset before testing on training dataset. During model training process, epoch accuracy *(val_acc)* should be increased and losses (val_loss) should be decreased. We have following possible cases for losses and accuracy:

1 If accuracy *(val_acc)* starts decreasing and losses *(val_loss)* start increasing, it means model is cramming values, and it is not learning.
2 If accuracy *(val_acc)* starts increasing and losses *(val_loss)* also starts increasing, it also means model is cramming values and not learning. This could be due to overfitting or diverse probability.
3 If accuracy *(val_acc)* starts increasing and losses *(val_loss)* start decreasing, it means model built is learning and is working fine.

Refer Fig. 10a–h for relationship between training and validation accuracy; and training and validation losses for a total of 50 epochs of each model.

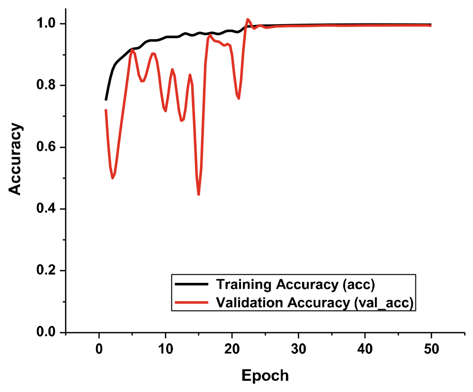
5.5 Prediction

After training the models, image has to be checked to predict the disease. For image checking, we use prediction python library "**Prediction.ImagePrediction**" from ImageAI. In this, we need to configure the JSON file path, which was generated during the training process, to specify the number of classes of healthy or diseased leaf images. As already shown in Fig. 10, we have achieved maximum accuracy for prediction using InceptionV3 model. But if we talk about prediction accuracy, then the DenseNet model has the highest prediction accuracy among all the models for selected images of diseased and healthy leaves. Comparisons of accuracy for test-results of different models are given in Table 4 for healthy leaves and leaves affected by Septoria leaf spot disease. Detailed comparisons of test results of different models are given in Table 5 for healthy leaves and leaves affected by various diseases.

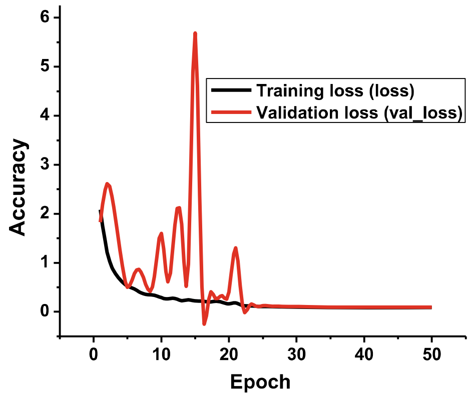
6 Conclusion

In this paper, we have trained datasets on different CNN models and have demonstrated the efficacy of all selected models using tomato leaf diseases images data for categorization. These CNN based approaches identify the distinctive features of the tomato leaf pictures and detect the nine common forms of tomato leaf diseases. Throughout this learning, to test different models and to achieve satisfactory result for generalization, a total of 18,160 images with uniform and complex background are used. We then trained the models using four different types of CNN algorithms viz. DenseNet, ResNet50, InceptionV3 and SqueezeNet for categorizing nine types of tomato plant diseases. We have used ImageAI libraries and classes for training and prediction on Google Colab to train and test accuracies of these models. Our experiments demonstrate a significant pull off in the prediction of accuracy for leaf disease using InceptionV3 as compared to other three models. We conclude that the best suitable model is InecptionV3 with the accuracy of 99.64%. The results demonstrate that the process will identify the nine common forms of tomato leaf diseases

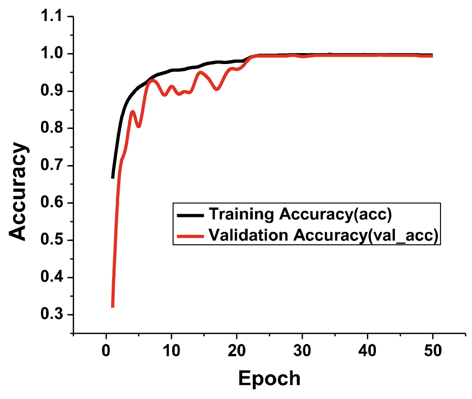
Fig. 10 a–h Training and validation accuracy and losses graph of all chosen models



(a) Training and Validation accuracy of DenseNet

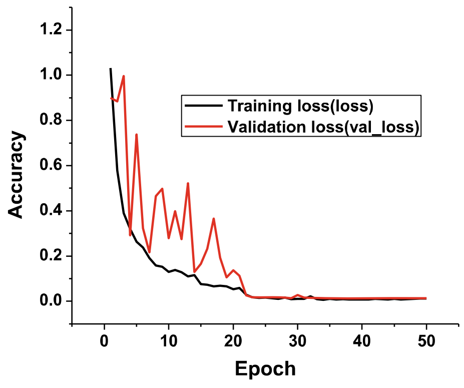


(b) Training and Validation loss of DenseNet

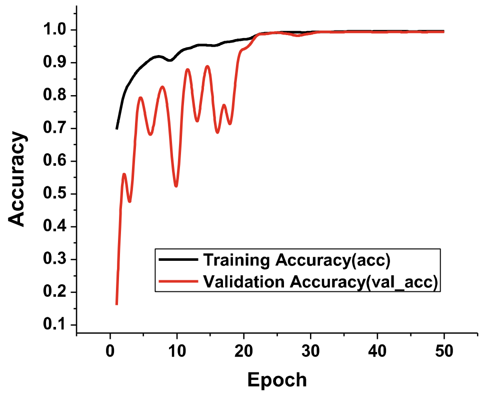


(c) Training and Validation accuracy of InceptionV3

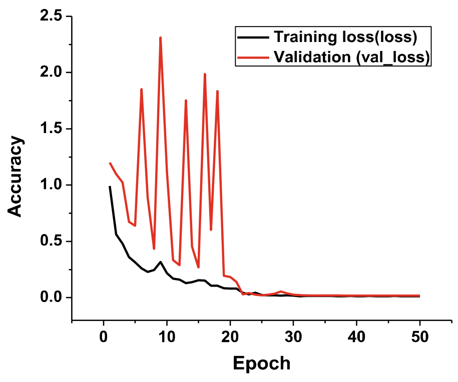
Fig. 10 (continued)



(d) Training and Validation loss of InceptionV3

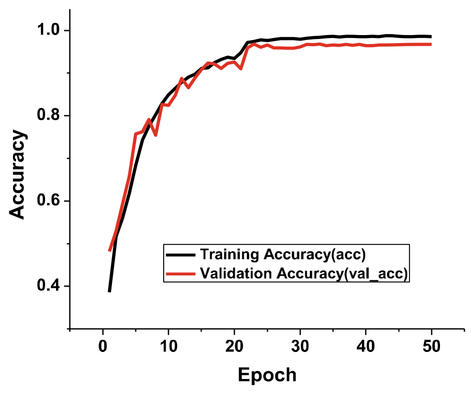


(e) Training and Validation accuracy of ResNet50

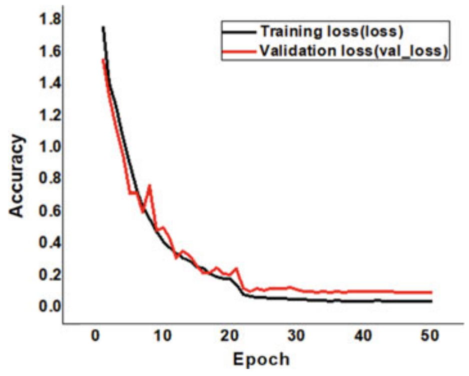


(f) Training and Validation loss of ResNet50

Fig. 10 (continued)



(g) Training and Validation accuracy of SqueezNet



(h)T raining and Validation loss of SqueezNet

Table 4 Model testing results on healthy and diseased leaves

S. No.	Model type	Test result of healthy leaf accuracy	Test result of Septoria Leaf Spot Disease accuracy
1	ResNet50	99.9983	99.9966
2	SqueezeNet	99.9753	99.9809
3	InecptionV3	99.9909	99.9980
4	DenseNet	99.9991	99.9997

with high accuracy. In future, we shall create our own dataset and build new CNN models for disease identification in other locally relevant crops like Soybeans, Jowar and Groundnut.

Table 5 Comparison of test-results of different models

Leaves status	Models and their test results			
	ResNet50	DensNet	InceptionV3	SqueezeNet
Healthy	99.9986	99.9992	99.9914	99.9771
Bacterial_spot	98.8272	98.1760	95.5207	99.8872
Early_blight	96.6901	90.6388	99.7619	99.4452
Late_blight	99.9963	99.9992	99.9999	99.9997
Leaf_Mold	99.9998	99.9998	99.9310	95.7853
Mosaic_virus	99.9999	99.9968	99.9996	99.9998
Septoria_leaf_spot	99.9933	99.9995	99.9930	99.9784
Two-spotted_spider_mite	99.9992	99.9995	99.6819	98.1829
Target_Spot	90.1588	95.6996	90.8067	99.0921
Yellow_Leaf_Curl_Virus	99.9920	99.9993	100.0000	100.0000

References

1. A.-K. Mahlein, T. Rumpf, P. Welke et al., Development of spectral indices for detecting and identifying plant diseases. Remote Sens. Environ. **128**, 21–30 (2013). https://doi.org/10.1016/j.rse.2012.09.019
2. A. Kamilaris, F.X. Prenafeta-Boldú, Deep learning in agriculture: A survey. Comput. Electron. Agric. **147**, 70–90 (2018). https://doi.org/10.1016/j.compag.2018.02.016
3. D. Ashourloo, H. Aghighi, A.A. Matkan et al., An investigation into machine learning regression techniques for the leaf rust disease detection using hyperspectral measurement. IEEE J. Sel. Topics Appl. Earth Obs. Remote Sens. **9**, 4344–4351 (2016). https://doi.org/10.1109/jstars.2016.2575360
4. Y. Lecun, Y. Bengio, G. Hinton, Deep learning. Nature **521**, 436–444 (2015). https://doi.org/10.1038/nature14539
5. M.M. Ghazi, B. Yanikoglu, E. Aptoula, Plant identification using deep neural networks via optimization of transfer learning parameters. Neurocomputing **235**, 228–235 (2017). https://doi.org/10.1016/j.neucom.2017.01.018
6. J. Carranza-Rojas, H. Goeau, P. Bonnet et al., Going deeper in the automated identification of Herbarium specimens. BMC Evol. Biol. (2017). https://doi.org/10.1186/s12862-017-1014-z
7. A. Caglayan, A.B. Can, Volumetric object recognition using 3-D CNNs on depth data. IEEE Access **6**, 20058–20066 (2018). https://doi.org/10.1109/access.2018.2820840
8. H. Lu, Y. Li, T. Uemura et al., Low illumination underwater light field images reconstruction using deep convolutional neural networks. Futur. Gener. Comput. Syst. **82**, 142–148 (2018). https://doi.org/10.1016/j.future.2018.01.001
9. S.P. Mohanty, D.P. Hughes, M. Salathé, Using deep learning for image-based plant disease detection. Front. Plant Sci. (2016). https://doi.org/10.3389/fpls.2016.01419
10. Y. Lu, S. Yi, N. Zeng et al., Identification of rice diseases using deep convolutional neural networks. Neurocomputing **267**, 378–384 (2017). https://doi.org/10.1016/j.neucom.2017.06.023
11. U.P. Singh, S.S. Chouhan, S. Jain, S. Jain, Multilayer convolution neural network for the classification of mango leaves infected by anthracnose disease. IEEE Access **7**, 43721–43729 (2019). https://doi.org/10.1109/access.2019.2907383
12. J.G. Barbedo, Factors influencing the use of deep learning for plant disease recognition. Biosys. Eng. **172**, 84–91 (2018). https://doi.org/10.1016/j.biosystemseng.2018.05.013

13. P. Jiang, Y. Chen, B. Liu et al., Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks. IEEE Access **7**, 59069–59080 (2019). https://doi.org/10.1109/access.2019.2914929
14. X. Zhang, Y. Qiao, F. Meng et al., Identification of maize leaf diseases using improved deep convolutional neural networks. IEEE Access **6**, 30370–30377 (2018). https://doi.org/10.1109/access.2018.2844405
15. K.P. Ferentinos, Deep learning models for plant disease detection and diagnosis. Comput. Electron. Agric. **145**, 311–318 (2018). https://doi.org/10.1016/j.compag.2018.01.009
16. A. Ramcharan, K. Baranowski, P. McCloskey, B. Ahmed, J. Legg, D.P. Hughes, Deep learning for image-based cassava disease detection. Front. Plant Sci. **8**, 1852 (2017)
17. J. Ma, K. Du, F. Zheng et al., A recognition method for cucumber diseases using leaf symptom images based on deep convolutional neural network. Comput. Electron. Agric. **154**, 18–24 (2018). https://doi.org/10.1016/j.compag.2018.08.048
18. A. Fuentes, S. Yoon, S. Kim, D. Park, A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition. Sensors **17**, 2022 (2017). https://doi.org/10.3390/s17092022
19. S. Kumar, B. Sharma, V.K. Sharma, H. Sharma, J.C. Bansal, Plant leaf disease identification using exponential spider monkey optimization. Sustain. Comput.: Inform. Syst. **28** (2018). https://doi.org/10.1016/j.suscom.2018.10.004
20. T.O. Emmanuel, PlantVillage Dataset, in Kaggle (2018). https://www.kaggle.com/emmarex/plantdisease. Accessed 18 Dec 2019
21. OlafenwaMoses, OlafenwaMoses/ImageAI, in GitHub (2018). https://github.com/OlafenwaMoses/ImageAI. Accessed 19 Nov 2019
22. J. Reddy, et al., Tomato farming project report, cultivation economics, in *Agri Farming* (2019). https://www.agrifarming.in/tomato-farming-project-report-cultivation-economics. Accessed 20 Nov 2019
23. K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016). https://doi.org/10.1109/cvpr.2016.90
24. H. Lee, I. Ullah, W. Wan et al., Real-time vehicle make and model recognition with the residual SqueezeNet architecture. Sensors **19**, 982 (2019). https://doi.org/10.3390/s19050982
25. S.-H. Tsang, Review: Inception-v3—1st Runner Up (Image Classification) in ILSVRC 2015, in *Medium* (2019). https://medium.com/@sh.tsang/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c. Accessed 25 Jan 2020
26. I. Khlestov, Notes on the Implementation of DenseNet in TensorFlow, in *Medium* (2017). https://medium.com/intuitionmachine/notes-on-the-implementation-densenet-in-tensorflow-beeda9dd1504. Accessed 25 Jan 2020
27. G. Huang, Z. Liu, L.V.D. Maaten, K.Q. Weinberger, Densely connected convolutional networks, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017). https://doi.org/10.1109/cvpr.2017.234