Computer Science • 23(3) 2022 https://doi.org/10.7494/csci.2022.23.3.4376

Subhash Mondal
Suharta Banerjee
Subinoy Mukherjee
Diganta Sengupta

PLANT DISEASE DETECTION USING ENSEMBLED CNN FRAMEWORK

Abstract *Agriculture exhibits the prime driving force for the growth of agro-based economies globally. In agriculture, detecting and preventing crops from the attacks of pests is a primary concern in today's world. The early detection of plant disease becomes necessary in order to avoid the degradation of the yield of crop production. In this paper, we propose an ensemble-based convolutional neural network (CNN) architecture that detects plant disease from the images of a plant's leaves. The proposed architecture considers CNN architectures like VGG-19, ResNet-50, and InceptionV3 as its base models, and the prediction from these models is used as an input for our meta-model (Inception-ResNetV2). This approach helped us build a generalized model for disease detection with an accuracy of 97.9% under test conditions.*

Keywords Convolutional Neural Network, Disease Detection, ResNet-50, VGG-19, InceptionV3

Citation Computer Science 23(3) 2022: 323–335

1. Introduction

Agriculture forms a primitive source of livelihood for agro-based economies. Industrialization stages throughout history, compounded by population progression, have generated a huge demand for agriculture. According to a report by the World Bank that was published in 2014, the agricultural sector contributes to one-third of the global gross domestic product (GDP). Agriculture is the primary source of income for more than 58% of the Indian population [16]. Although technological advancements had shifted the focus from agriculture to other engagement options, the rise of artificial intelligence (AI) has again restored the focus in agriculture with the aid of AI tools in the last decade. This paper concentrates on the use of AI tools for agricultural purposes – most definitely in plant disease prediction. The functionality of the framework demands that crops are to be very closely monitored for detecting any type of disease in the initial stage before harvesting. This will help us minimize crop losses by applying pesticides if diseases are correctly diagnosed and identified early. Generally, the diseases of plants include bacteria, molds, viruses, fungi, etc. These diseases are generally identified by farmers or experts through human vision over longer periods [13]. Nevertheless, this can also be expensive and inaccurate, thereby mandating the use of deep learning (DL) for the detection and classification of plant diseases. Over the past few years, DL has been gaining popularity and importance due to its performance results in accuracy when trained with a huge dataset. This work employs DL to build an automated framework that is capable of accurately predicting a specific plant disease. An ensemble-based convolutional neural network (CNN) approach is proposed that uses transfer-learning techniques that help in building a generalized disease-detection model for different plants. Three different state-of-the-art models (VGG-19, ResNet-50, and InceptionV3) have been used as a base model for our proposed architecture. The proposed ensemble technique helped us achieve higher accuracy in contrast to a single standalone model.

The rest of the paper is organized as follows: Section 2 deals with a literature survey, while Section 3 illustrates our proposed architecture. Our experimental results & analysis are outlined in Section 4, and the paper concludes with Section 5.

2. Literature Survey

Recently, several DL technologies have been used for detecting diseases in plants. A different architecture has been introduced with a different efficiency in order to improve their results. The larger amount of data availability has helped to improve the classification performance of DL techniques for computer vision problems. The choice of architecture can vary; however, it is important to understand the different architectures to apply them effectively. The authors of [11] used the transfer-learning technique on the PlantVillage dataset; its accuracy was recorded to be approximately 90.4%. The authors of [4] used two different techniques for testing the presence of two specific diseases; the model detected spot diseases and diseases caused by fungus. They

used HSI images instead of RGB images; when they used RGB, the leaf veins degraded the performance (as there was excessive noise in the images), but HSI images reduced the noise that was present due to the leaf veins. The model was also able to detect the intensity of fungus attacks by measuring some parameters. The main disadvantage of the model was that it classified the training plants that had diseases other than spot and fungus as healthy. The authors of [8] introduced a system for detecting the different diseases that are encountered by tomato plants. The primary aim of the introduced system was to use the easiest method to detect a particular disease as well as to involve a minimal use of computational resources to produce the results. This approach proved to be quite effective, showing an accuracy of approximately 95%. The system was built by making some changes in the CNN LeNet design. Another proposed method in [15] detected diseases of rice. The disease-recognition system was built based on a CNN. The system tested around 500 pictures of rice leaves and stems; after the training, ten common diseases were classified. Many different studies have been done specifically on rice leaves; for example, [1] used an artificial neural network (ANN) to detect the symptoms of rice blast disease in rice plants. Burhan et al. [2] used five different models (VGG16, VGG19, ResNet50, ResNet101V2, and ResNet50V2) and compared their performance levels. The models tested on both artificial data as well as real images that were collected from fields.

3. Proposed Methodology

In this paper, we propose an ensemble-based DL approach for detecting plant disease using a state-of-the-art CNN. We have designed a disease-detection pipeline (Figure 1) that is responsible for automating the task of disease identification and providing prevention methodologies using computer vision and machine-learning techniques.

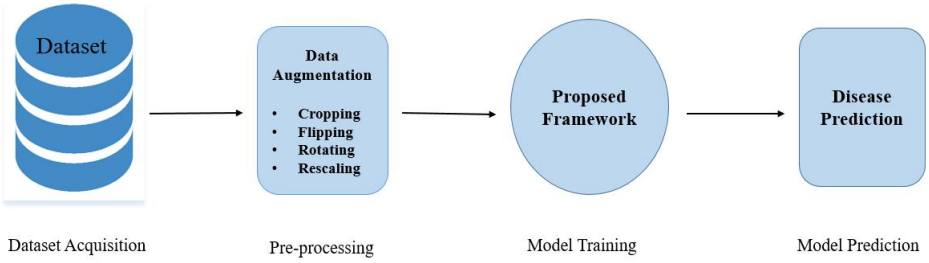

Figure 1. Disease-detection pipeline

3.1. Dataset Acquisition

This step is responsible for collecting and organizing the images for our database (which will serve as our input in the pipeline). Those images that belonged to a

particular class were grouped, and the name of each folder later served as our class names (Table 1) during the training process in our pipeline. We have considered two plants (namely, tomato and cotton) and have studied the diseases that are associated with them in order to build a generalized pipeline for disease identification that can be used for any plant. We used the tomato leaf disease detection dataset (Figure 2), which is available in [10], for building our proposed architecture. This dataset consists of 18,345 images that belong to 10 classes (as presented in Table 1).

Table 1
Class Name of Tomato & Cotton Dataset

Tomato Dataset		Cotton Dataset	
Class	Class Name	Class	Class Name
1	Bacterial spot	1	Bacterial blight
2	Early blight	2	Curl virus
3	Late blight	3	Fussarium wilt
4	Leaf mold	4	Healthy
5	Septoria leaf spot		
6	Spider mite/two-spotted spider mite		
7	Target spot		
8	Tomato yellow leaf curl virus		
9	Tomato mosaic virus		
10	Healthy		

Out of the total number of available images, 1926 images were of healthy tomato leaves, and the rest depicted different diseases like bacterial spot, early blight, late blight, leaf mold, Septoria leaf spot, and some other relevant diseases. Our proposed architecture was further exposed to the cotton leaf disease detection dataset (Figure 2), which is available in [9], to build a more generalized system for classifying images with higher accuracy. The dataset consists of 1711 images that belong to 4 classes. Diseases that are associated with cotton leaves include bacterial blight, curl virus, and Fusarium wilt. The organized images from these datasets were then passed on to the pre-processing module of our proposed pipeline in order to carry out various pre-processing mechanisms that helped us improve the accuracy and performance of our proposed ensemble-based architecture.

Figure 2. Dataset sample

3.2. Pre-processing Module

In this module, various image pre-processing techniques were applied, which contributed to better model performance. As a part of the image pre-processing, we used the technique of data augmentation. Data augmentation helps in improving the amount of data that is available during training and also incorporates diversity into our training data, which helped us overcome the problem of overfitting. Different types of transformations (as shown in Figures 3 and 4) were applied to each image at each epoch during the runtime; hence, additional memory space was not required.

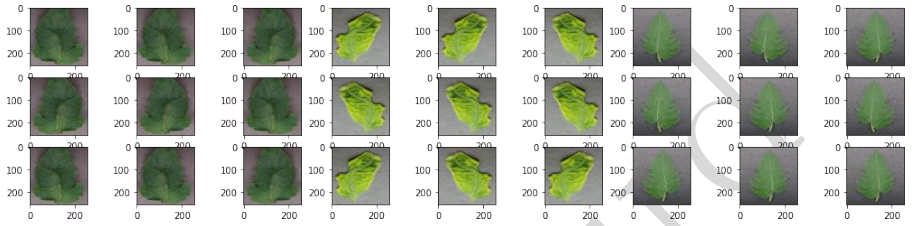

Figure 3. Flip augmentation

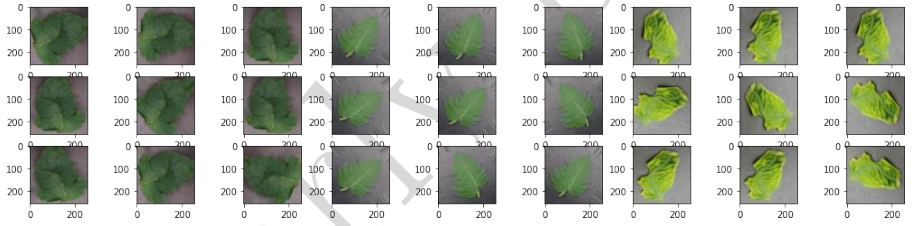

Figure 4. Rotation augmentation

The various pre-processing steps included reducing the sizes of the images, cropping the images, flipping, rotating, rescaling, and many other steps that helped us create variances that led toward building a stable model. The researchers revealed that a gray-scale image does not provide us with high performance when compared to RGB images [12] [3]. As a matter of fact, our proposed framework takes color images into account that are resized to a resolution of 224×224 for the further model-building steps.

3.3. Proposed Framework

In our proposed framework (Figure 5), we used an ensemble-based computer-vision approach for identifying the types of diseases that were encountered by a plant based on the images of the leaves of the plant. The ensemble technique uses the idea of aggregating the predictions from a group of predictors, which helped us achieve better predictions as compared to the use of a single predictor.

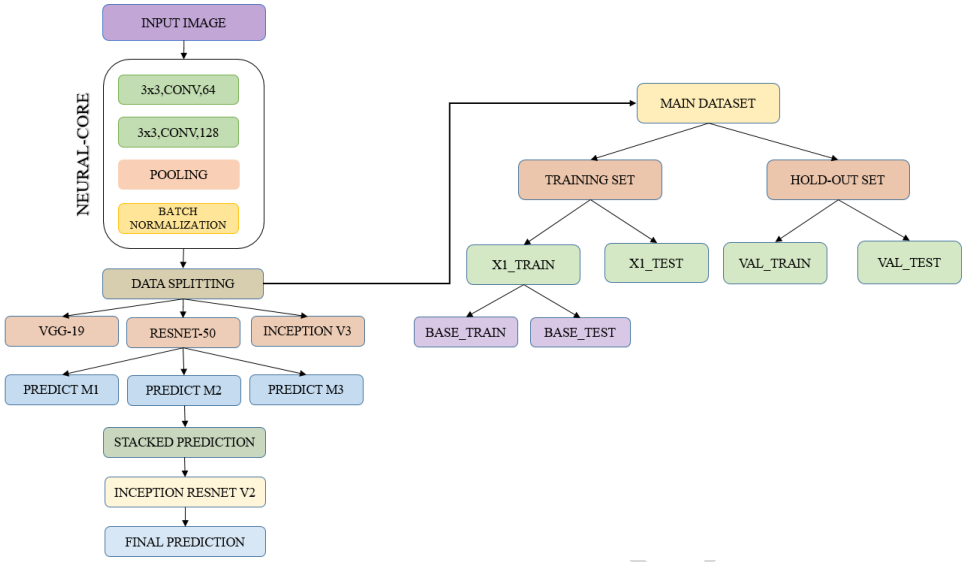

Figure 5. Proposed framework

The images that were obtained from the pre-processing module of this detection pipeline served as the input for our architecture. The images were first fed into the neural core of our architecture. Inside this neural core, we implemented a double 3×3 convolutional layer, followed by pooling and batch normalization. The convolutional layer helped us extract various features from the images and also helped us learn the changes in the weights and biases of the different feature maps. The filters (which are also considered to be learnable parameters) were moved across the entire image in order to obtain a feature map. After the convolutional layers, we applied activation functions. The proposed methodology was tested using four different activation functions; namely, *tanh*, *softmax*, *sigmoid*, and *ReLu*. The *ReLu* activation function helped us to normalize and non-linearize our dataset. The *ReLu* activation function was chosen because it is a differential function, and it overcomes the problem of a vanishing gradient. A comparison of the activation functions is shown in Table 2.

Table 2
Comparison of activation functions

Activation Functions	Accuracy
Tanh	82.3
Softmax	93.6
Sigmoid	94.7
ReLu	97.9

The pooling layer helped us reduce the dimensionality of our data. A certain amount of regularization was added by using batch normalization; this normalized all of the data points into a range of between 0 and 1 [7], which drastically reduced the time that was required for computation. The whole idea of introducing a neural core into our architecture was to efficiently extract features from the images for building a generalized model. A comparison study is shown in Table 3. Figures 6 and 7 present the hidden Deep-CNN layers for the three approaches.

Table 3
Comparison study with/without neural core

	Accuracy	Loss
Proposed Framework without Neural Core	95.2	1.2098
Proposed Framework with Neural Core	97.9	0.7832

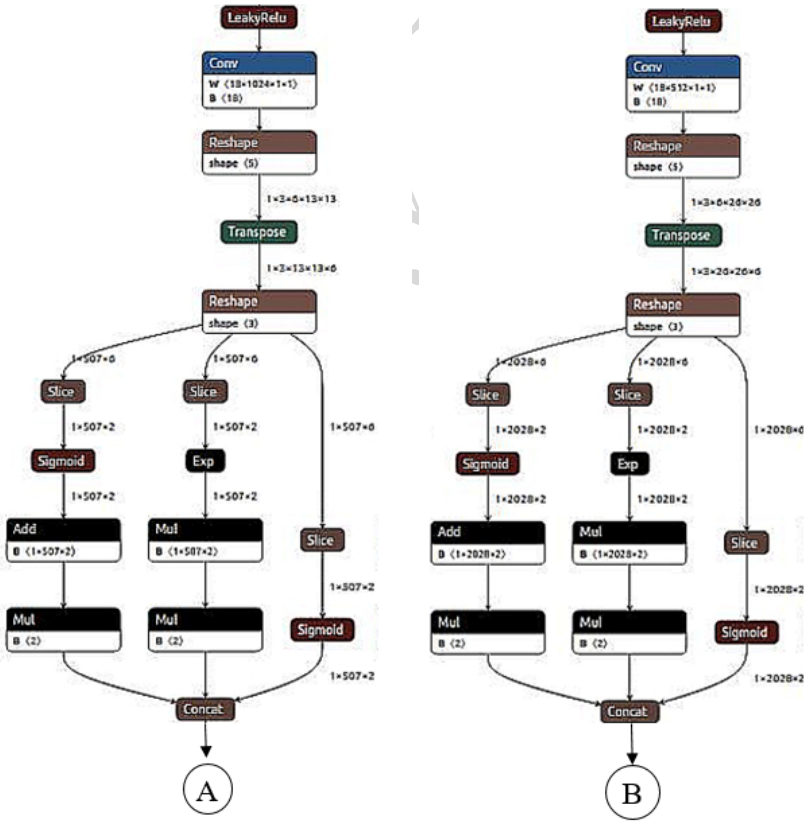

Figure 6. Deep CNN hidden layers

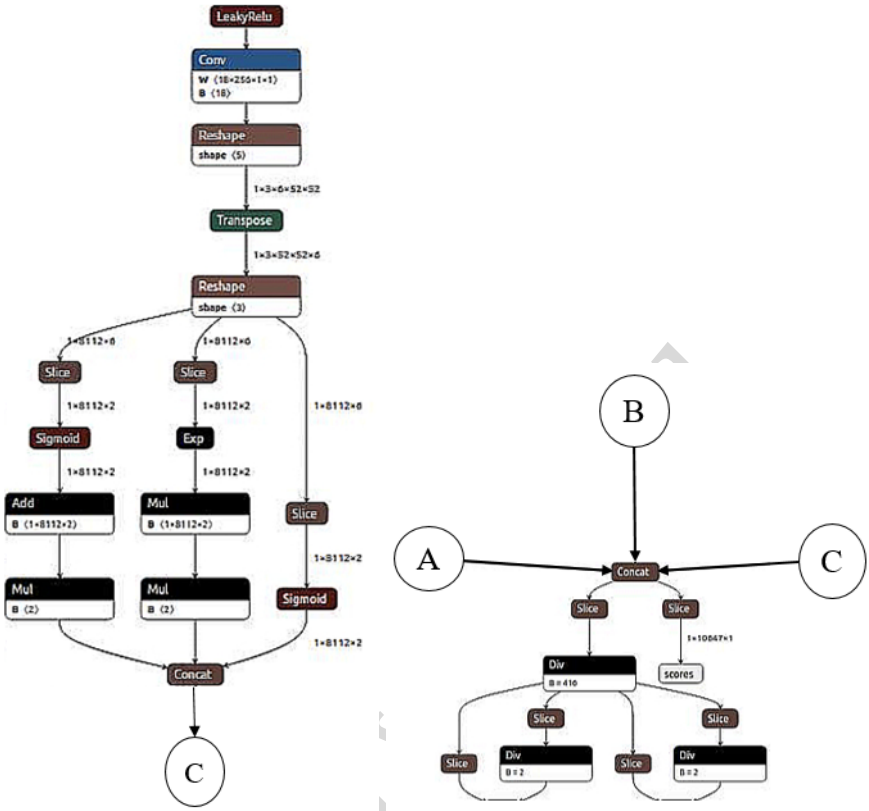

Figure 7. Deep CNN hidden layers

Our model was trained by using three different algorithms (namely, VGG-19, ResNet- 50, and InceptionV3 [5, 8, 14]), which served as the base models for our ensemble-based approach. In our proposed methodology, we used the process of stacking in order to aggregate our predicted results. *Stacking* is an ensemble technique that helps in improving performance results. In this process, different models are trained on the same dataset. The architecture consists of two or more models referred to as a level-0 model and one final model referred to as a level-1 model. Level-0 model or base model is fitted on training data, and its prediction is compiled, while the level-1 model or meta-model is responsible for learning how to combine the prediction of the base model in the best possible way.

The dataset splitting was achieved by using the seven-fold cross-validation technique (which helped us prevent the problem of model overfitting). The entire dataset was split into two different parts: the training set, and the hold-out set. The training set was split into $X1_{Train}$ and $X1_{Test}$, while the hold-out set was split into Val_{Train} and Val_{Test}. $X1_{Train}$ was further split into two parts: $Base_{Train}$, and $Base_{Test}$ –

these were used for training our base models. After the training of these base models, the predictions from each were obtained by using the Val_{Train} dataset. These predictions were stacked together using the average of all of the predictions (which will serve as input for the meta-model). Along with these stacked predictions, the Val_{Test} dataset was used for training our meta-model that used the Inception-ResNetV2 architecture. The prediction in the meta-model was made by using the $X1_{Test}$ dataset, and the accuracy of the model was calculated accordingly.

4. Experimental Results and Analysis

The confusion matrix helps us to determine the performance of various learning algorithms; it provides us with a comparison of the actual and predicted values. Generally, a confusion matrix provides us with four different terms: namely, a true positive (TP), false positive (FP), true negative (TN), and false negative (FN). With the help of these terms, we can deduce the precision and recall, which helps us understand how good or bad our model is performing.

In the case of multi-class classification problems, however, the generated confusion matrix does not directly give us the values of TP, FP, TN, and FN (as in the case of a binary classification). In our problem, we had four classes in the cotton dataset, and the generated confusion matrix was a 4×4 matrix. Here, Cell BB represents the TP value of the bacterial blight class. FN was presented by the sum of all of the corresponding rows except for TP; i.e., the FN for bacterial blight was $2 \times (cellBC + cellBF + cellBH)$. The FP was depicted by all of the corresponding columns except for TP; for bacterial blight, the FP value was $3 \times (cellCB + cellFB + cellHB)$. Similarly, these values were calculated for the other classes; with the help of these, the obtained precision and recall were presented in the subsequent section. Figure 8 represents the confusion matrix that was obtained for the cotton dataset.

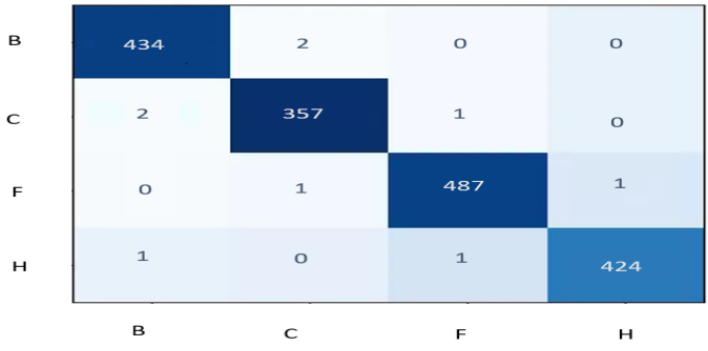

Figure 8. Confusion matrix: B – bacterial blight; C – curl virus; F – Fussarium wilt; H – healthy

4.1. Comparative Analysis

The use of stacking (which is a part of the ensemble technique) helped us improve the accuracy of the model by combining the predictions of the base models and using them as the input for our meta-model. In our framework, we used the concept of transfer learning for building the model by using VGG-19, ResNet-50, and InceptionV3 as the base model of our architecture. The optimal hyperparameter was chosen with the help of the Keras tuner. The optimizer that was used was the *Adam optimizer*, and the loss function that was used for the entire experiment was *categorical loss entropy*. The approach was compared with standard standalone architectures such as AlexNet, VGG-16, and ResNet-50. The proposed approach helped us build a more generalized detection system that was able to detect and classify the diseases with a very low error rate. The performance analysis when using the tomato & cotton datasets is shown in Table 4. A comparison of the different approaches in the literature is shown in Table 5. The accuracy, precision, and recall comparisons are shown in Figure 9.

Table 4

Performance analysis using tomato & cotton datasets: Arch – architecture; Acc – accuracy; Prec – precision; Rec – recall

Tomato Dataset				Cotton Dataset				Algorithm Ref No.
Arch	Acc	Prec	Rec	Arch	Acc	Prec	Rec	
AlexNet	72.8	0.76	0.72	AlexNet	75.3	0.84	0.78	S.A.3
VGG16	89.4	0.88	0.82	VGG16	92.3	0.88	0.83	S.A.5
ResNet50	94.3	0.85	0.91	ResNet50	96.1	0.91	0.87	S.A.4
Proposed	97.6	0.89	0.96	Proposed	98.2	0.95	0.94	S.A.8

Table 5

Comparative analysis

Proposal	VGG	ResNet	InceptionV3	Inception ResNetV2	Data Augmentation	Neural Core
[11]	X	✓	X	X	✓	X
[4]	✓	X	X	X	X	X
[15]	✓	X	X	X	✓	X
[1]	✓	X	X	X	X	X
[6]	X	✓	✓	X	✓	X
Proposed Framework	✓	✓	✓	✓	✓	✓

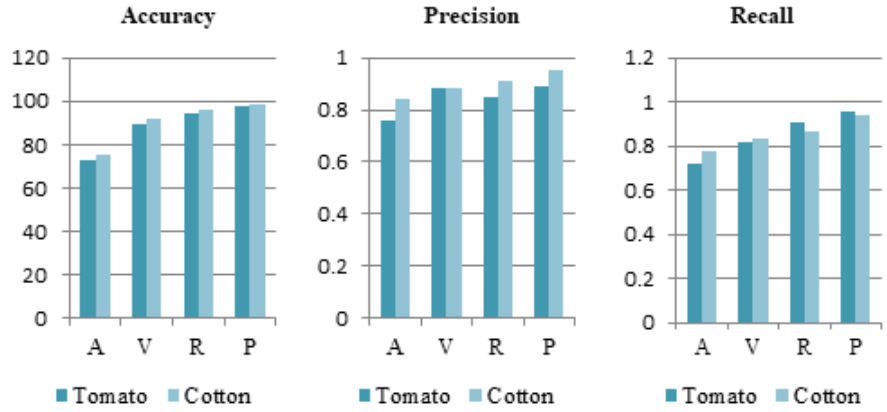

Figure 9. Accuracy, precision, and recall: A – AlexNet; V – VGG; R – ResNet50; P – proposed

5. Conclusion

In this paper, we proposed an ensemble-based architecture that can classify plant diseases. We used VGG-19, Resnet-50, and Inception V3 as our base model for classifying various diseases using the transfer-learning technique. We tested our approach using two different datasets, which helped us build a generalized model for plant disease detection. We employed pre-processing techniques like data augmentation to ensure that our model did not overfit. Our proposed approach successfully classified diseases with accuracies of 97.6% for the tomato dataset and 98.2% for the cotton dataset. In the future, we plan to incorporate image segmentation techniques along with our proposed CNN technique, which will not only help us classify the diseases but also identify their areas of attack.

References

[1] Andrianto H., Suhardi, Faizal A., Armandika F.: Smartphone Application for Deep Learning-Based Rice Plant Disease Detection. In: *2020 International Conference on Information Technology Systems and Innovation (ICITSI)*, pp. 387–392, 2020. doi: 10.1109/ICITSI50517.2020.9264942.

[2] Burhan S.A., Minhas S., Tariq A., Nabeel Hassan M.: Comparative Study Of Deep Learning Algorithms For Disease And Pest Detection In Rice Crops. In: *2020 12th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pp. 1–5, 2020. doi: 10.1109/ECAI50035.2020.9223239.

[3] Ferentinos K.P.: Deep learning models for plant disease detection and diagnosis, *Computers and Electronics in Agriculture*, vol. 145, pp. 311–318, 2018. doi: https://doi.org/10.1016/j.compag.2018.01.009.

[4] Habiba S.U., Islam M.K.: Tomato Plant Diseases Classification Using Deep Learning Based Classifier From Leaves Images. In: *2021 International Conference on Information and Communication Technology for Sustainable Development (ICICT4SD)*, pp. 82–86, 2021. doi: 10.1109/ICICT4SD50815.2021.9396883.

[5] He K., Zhang X., Ren S., Sun J.: Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[6] Huang S., Liu W., Qi F., Yang K.: Development and Validation of a Deep Learning Algorithm for the Recognition of Plant Disease. In: *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 1951–1957, 2019. doi: 10.1109/HPCC/SmartCity/DSS.2019.00269.

[7] Ioffe S., Szegedy C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, p. 448–456, ICML'15, JMLR.org, 2015.

[8] Jasim M.A., AL-Tuwaijari J.M.: Plant Leaf Diseases Detection and Classification Using Image Processing and Deep Learning Techniques. In: *2020 International Conference on Computer Science and Software Engineering (CSASE)*, pp. 259–265, 2020. doi: 10.1109/CSASE48920.2020.9142097.

[9] Karim S.: Kaggle: cotton leaf disease dataset. https://www.kaggle.com/seroshkarim/cotton-leaf-disease-dataset.

[10] Lamrahi N.: Kaggle: New Plant Diseases Dataset(Augmented). https://www.kaggle.com/noulam/tomato.

[11] Marzougui F., Elleuch M., Kherallah M.: A Deep CNN Approach for Plant Disease Detection. In: *2020 21st International Arab Conference on Information Technology (ACIT)*, pp. 1–6, 2020. doi: 10.1109/ACIT50332.2020.9300072.

[12] Mohanty S.P., Hughes D.P., Salathé M.: Using Deep Learning for Image-Based Plant Disease Detection, *Frontiers in Plant Science*, vol. 7, 2016. doi: 10.3389/fpls.2016.01419.

[13] Narvekar P., Patil S.: Novel algorithm for grape leaf diseases detection, *International Journal of Engineering Research and General Science*, vol. 3(1), pp. 1240–1244, 2015.

[14] Szegedy C., Vanhoucke V., Ioffe S., Shlens J., Wojna Z.: Rethinking the inception architecture for computer vision. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[15] Wang Q., He G., Li F., Zhang H.: A novel database for plant diseases and pests classification. In: *2020 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, pp. 1–5, 2020. doi: 10.1109/ICSPCC50002.2020.9259502.

[16] Yashwanth M., Chandra M.L., Pallavi K., Showkat D., Kumar P.S.: Agriculture Automation using Deep Learning Methods Implemented using Keras. In: *2020 IEEE International Conference for Innovation in Technology (INOCON)*, pp. 1–6, 2020. doi: 10.1109/INOCON50539.2020.9298415.

Affiliations

Subhash Mondal
Department of Computer Science & Engineering, Meghnad Saha Institute of Technology, Kolkata, India 700150, e-mail: subhash@msit.edu.in

Suharta Banerjee
Department of Computer Science & Engineering, Meghnad Saha Institute of Technology, Kolkata, India 700150, e-mail: suharta˙b.cse2018@msit.edu.in

Subinoy Mukherjee
Department of Computer Science & Engineering, Meghnad Saha Institute of Technology, Kolkata, India 700150, e-mail: subinoy˙m.cse2018@msit.edu.in

Diganta Sengupta
Department of Computer Science & Engineering, Meghnad Saha Institute of Technology, Kolkata, India 700150.
Department of Computer Science & Business Systems, Meghnad Saha Institute of Technology, Kolkata, India 700150, e-mail: sg.diganta@ieee.org

Received: 06.07.2021
Revised: 23.09.2021
Accepted: 08.07.2022