Convolutional neural network architecture for detection and classification of diseases in fruits

Yogesh Kumar, Nitasha Hasteer*, Anshul Bhardwaj and Yogesh

Amity University, Noida 201 313, India

Artificial intelligence is now becoming a part of people's everyday lives. It can help farmers detect any disease in the early stage and take pre-emptive actions to save their crops and control disease spread, thus preventing crop wastage as well as increasing their income. The present study uses a combination of 13 convolutional neural network (CNN) models to classify five types of fruits and their leaf images into 41 classes, including diseased and healthy. Results show that the average accuracy of this CNN architecture is above 90% for all 13 individual models. One of the CNN models has been compared with three pre-trained models, i.e. MobileNet, DenseNet121 and InceptionV3 trained using the same dataset. It shows that the CNN architecture used in this study has higher accuracy while also being simple and easy to train.

Keywords: Agriculture, artificial intelligence, convolutional neural network, deep learning, fruit and leaf disease detection.

THE convolutional neural network (CNN) or ConvNet is an algorithm in deep learning which takes an image as input and assigns weights to different parts of the image which help the algorithm to differentiate between different kinds of images. CNN imitates the working of human neurons and their connectivity. The architecture of CNN has been inspired by the design and working of the human visual cortex. Individual neurons can only respond to stimuli in a small part of the visual field called the receptive field. A number of similar fields are stacked on top of one another to span the full visual field. When compared with different classification methods, the amount of pre-processing required by CNN is significantly less which makes it ideal for such tasks.

India is an agriculture-dominant country, and majority of the population depends on agriculture or its related activities for their livelihoods. Every year several fruits are affected by diseases and thus wasted[1]. According to the Food and Agriculture Organization (FAO), Rome, up to 40% of the world's fruit crops are lost due to pests and diseases[2]. This loss could be reduced if the farmers could detect the disease early so that the crops could be treated and the fruits saved.

*For correspondence. (e-mail: nitasha78@gmail.com)

This study uses a combination of CNN models which can be directly accessed by the farmers, who can take a photograph of disease-affected leaf or fruit and upload it to the network. It will then identify the disease and thus help the farmers.

India is currently going through a digital revolution and nowadays many rural areas have internet access. CNN could be hosted on the web for easy access and can be directly used by the farmers to identify the diseases affecting their crops. If the farmers detect the disease early, remedial actions can be taken to save the fruits. Even if the disease has propagated in a part of the fruit crop, it would be beneficial to identify the same so that the disease can be stopped from spreading further.

Literature review

A lot of work is already being done in this field. Some of the literature from sources like Scopus, Springer, IEEE and ACM have been reviewed here. It also includes details about the current CNN architecture or models being used by researchers in this field.

James and Sujatha[3] proposed a Hybrid Neural Clustering (HNC) classifier to classify apple fruit images. They used *K*-means clustering and feed-forward backpropagation (FFBP) neural network to develop the classifier. The model could classify apple images into ten disease categories with an accuracy of 98% (ref. 3).

Sembiring *et al.*[4] proposed a CNN architecture for the classification of tomato leaves. Their main focus was to develop a simplified CNN architecture that could provide acceptable accuracy. These researchers used the PlantVillage dataset with ten classes consisting of nine diseases and one healthy class. They compared the accuracy of their proposed architecture with other CNN architectures like VGG Net, ShuffleNet and SequeezNet, which had achieved an accuracy of 97.15% (ref. 4).

Xiao *et al.*[5] proposed a CNN model to classify strawberry images into five disease categories. They used both fruit and leaf images to train their CNN model. They also compared their proposed model with GoogleNet, VGG16 and ResNet50. The accuracy of their model was 99.60% (ref. 5).

Ashwinkumar *et al.*[6] developed a CNN model based on MobileNet. The proposed OMNCNN model operates at

Table 1. Literature review of recent fruit disease detection works

Fruit	Method used	Accuracy achieved (%)	Reference
Pineapple	ANN, SVM, RF, NB, DT, KNN and ANOVA	85–90	9
Tomato	SqueezNet, AlexNet, Inception V3	89.69 and 93.4	10
Tomato	VGG, ResNet, and DenseNet	96.16	11
Strawberry	CNN	99.60	5
Tomato	CNN	98.28, 96.64 and 97.01	4
Apple	ResNet-50 using MASK RCNN and Transfer Learning	99.1	7
Cherry	Data Mining Prediction	95.8	12
Tomato	GLCM-Color Moment and CNN	99	13
Mango	DCNN and Transfer Learning	98.6	14
Apple	HNC Classifier, FFBPNN	98.1	3
Tomato	AlexNet, GoogleNet	99.18	15
Tomato	OMNCNN	Above 98	6

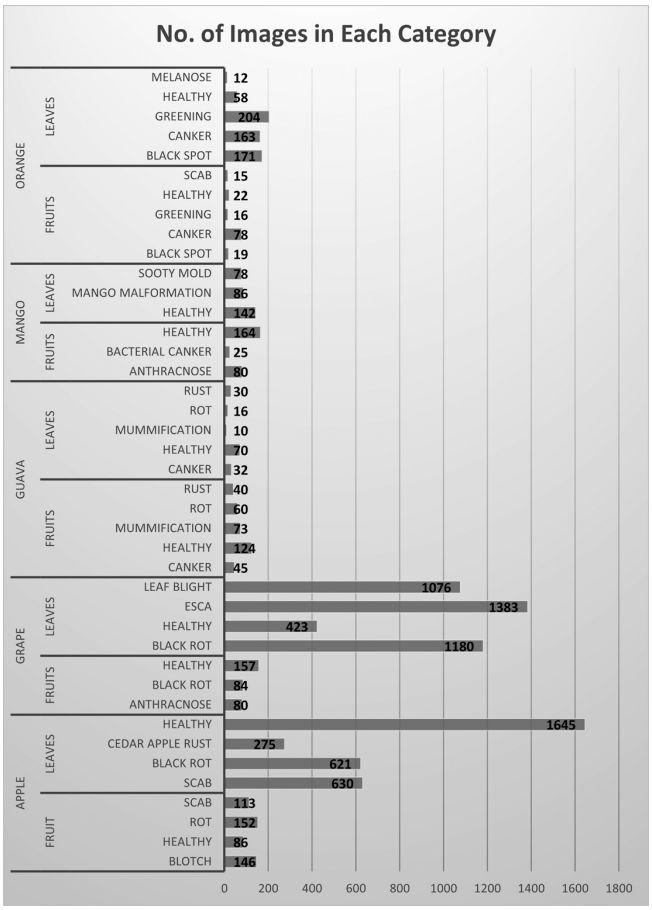

Figure 1. Number of images in the dataset.

multiple stages such as pre-processing, segmentation, feature extraction and classification. This model was trained using a dataset of tomato leaves. They used the emperor penguin optimizer (EPO) algorithm and extreme learning Machine (ELM)-based classifier techniques. Their model showed an accuracy of 98.7% (ref. 6).

Rehman *et al.*[7] proposed a MASK RCNN model to detect apple leaf diseases. Their model involved a novel, parallel, real-time processing framework and was trained using transfer learning. They also employed an ensemble subspace discriminant analysis (ESDA) classifier. Their model was trained using the PlantVillage dataset and achieved an accuracy of 96.6% (ref. 7).

Jogekar and Tiwari[8] proposed a deep convolutional neural network (DCNN) to classify banana diseases. They also proposed a genetic DCNN algorithm that could be used to minimize the loss of available data for some classification problems. They also developed a web application that could be used to detect banana leaf disease with visual inspection by the users. They used logistic regression, support vector machine, Gaussian Naïve Bayes and LDA techniques for their DCNN model. The accuracy achieved by their model was more than 90% (ref. 8). Table 1 lists some recent works in this area[9–15].

Methodology

Dataset

The dataset used to train the CNN models was collated by the present authors from various sources such as Kaggle, PlantVillage, Mendeley Datasets, Google images, etc. Data were cleaned and pre-processed before being used to train the classifiers. The dataset consisted of five fruits and their leaves. Totally there were 9890 images which had been segmented into two parts for each fruit and leaf as train and test.

Figure 1 shows the structure of the dataset and the number of images in each class.

Dataset augmentation

The images from the dataset were processed using the ImageDataGenerator function, where different attributes such as rescaling, zoom_range, sheer_range and horizontal_flip were applied to the training data. The images from the test dataset were only rescaled to 1./255, and no other filter was applied.

The datasets for test and train were generated by passing these images through flow_from directory, where class_mode

was taken as categorical and color_mode as rgb. All the images were resized to 500*500 before being provided as an input to the CNN. The batch_size for input images was taken as 16.

Tools and libraries used

We have used DataSpell, PyCharm and Google Colab for developing the classifiers. Python3 and its libraries TensorFlow[16], Keras[17], Scikit-learn[18], Pandas[19], Numpy[20], Matplotlib[21] and Seaborn were used for various tasks. The CNN models were trained on a laptop with Intel i7-9750H CPU and Nvidia GTX 1650 GPU with 16GB RAM. The models were trained in a CUDA-enabled environment using DataSpell IDE. All models for comparison were trained in Google Colab.

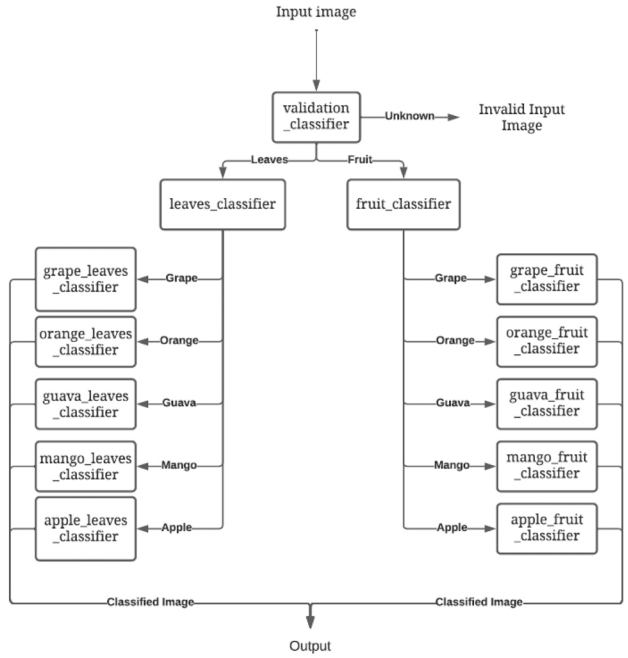

Figure 2. Control-flow diagram of the convolutional neural network (CNN) model.

Layer (type)	Output Shape	Param #
conv2d_23 (Conv2D)	(None, 498, 498, 32)	896
max_pooling2d_20 (MaxPooling	(None, 249, 249, 32)	0
conv2d_24 (Conv2D)	(None, 247, 247, 32)	9248
max_pooling2d_21 (MaxPooling	(None, 123, 123, 32)	0
conv2d_25 (Conv2D)	(None, 121, 121, 64)	18496
max_pooling2d_22 (MaxPooling	(None, 60, 60, 64)	0
conv2d_26 (Conv2D)	(None, 58, 58, 64)	36928
max_pooling2d_23 (MaxPooling	(None, 29, 29, 64)	0
conv2d_27 (Conv2D)	(None, 27, 27, 128)	73856
max_pooling2d_24 (MaxPooling	(None, 13, 13, 128)	0
flatten_4 (Flatten)	(None, 21632)	0
dense_16 (Dense)	(None, 128)	2769024
dense_17 (Dense)	(None, 64)	8256

Figure 3. Classifiers' structure till the last dense layer.

Proposed model

Here we propose a combination of 13 CNN models which have been trained using a self-collated dataset. The dataset includes five of the most common fruits in India, viz. apple, grapes, guava, mango and orange. Three out of these 13 models were used to classify each input image.

The first CNN model, viz. validation_classifier, was used for the validation of the input image. It classifies an image into three classes – leaves, fruit and unknown. If the image is classified as a fruit, it is transferred to the fruit_classifier; if classified as a leaf, it is transferred to the leaves_classifier, and if the image is classified as unknown, it gives a warning 'Invalid input image'. The fruit_classifier and leaves_classifier models classify the image into five classes. Based on the result from the fruit_classifier or leaves_classifier, the image is provided to the appropriate CNN model for disease identification. This result is then provided to the user. Figure 2 shows the control flow structure of the proposed model.

Implementation

The CNN architecture helps farmers identify the diseases that might be affecting their fruit crops. Using an image of a leaf or fruit, the network can classify it as healthy or diseased and also identify the particular disease.

Figure 4. Output image of the CNN model.

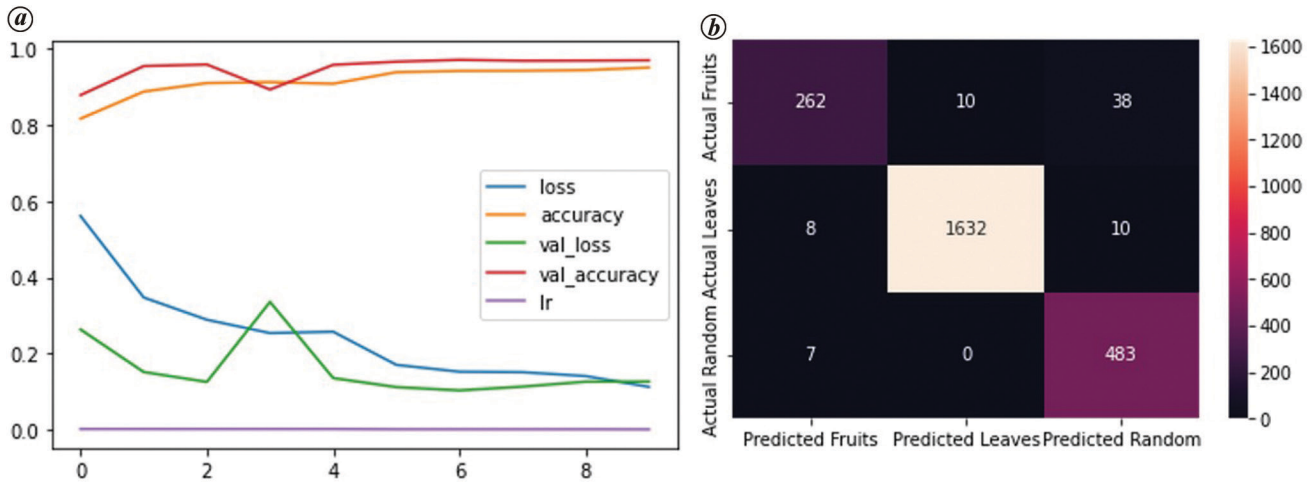

Figure 5. ***a***, Training history of validation_classifier. ***b***, Confusion matrix of validation_classifier.

Table 2. Accuracy of all CNN models

Model	Training accuracy (%)	Transfer accuracy (%)	No. of epochs
validation_classifier	95.42	97.02	9
fruit_classifier	98.40	83.10	27
leaves_classifier	96.70	95.89	15
apple_fruit_classifier	91.34	82.38	19
apple_leaves_classifier	98.27	95.10	6
grape_fruit_classifier	92.02	86.67	25
grape_leaves_classifier	99.75	98.64	15
guava_fruit_classifier	90.61	81.72	30
guava_leaves_classifier	93.50	84.52	24
mango_fruit_classifier	91.20	81.39	21
mango_leaves_classifier	90.42	90.00	8
orange_fruit_classifier	94.17	86.67	18
orange_leaves_classifier	93.97	82.40	28

CNN architecture and training

All the models have been individually trained using the whole or partial dataset. A common architecture has been used for all the models, with a few necessary tweaks. Figure 3 gives the structure of this CNN architecture without the final softmax layer.

The CNN architecture consists of 13 layers, of which five are convolution layers, five are pooling layers and three are fully connected layers. The first is a convolution layer with a kernel size of 3, 'relu' activation and unit as 32. Its input shape is defined as 500*500*3. No padding has been applied to input images in any of the layers. This layer is then connected to a pooling layer which uses Max Pooling with a size of 2*2. The output of this pooling layer is transferred to the next layer which is another convolution layer with the same parameters, except for the input_shape as the first layer and is further connected to an identical MaxPooling layer. The next two convolution layers also have the same kernel size and activation, however they have 64 units instead of 32 and are also connected to another exactly same pooling layer. The next is the last convolution layer of the CNN architecture having 128 units, and is also succeeded by a MaxPooling layer. After all these convolution and pooling layers, the output is flattened and provided to a fully connected layer with 128 units and 'relu' activation. The output of this layer is further provided to the next fully connected layer with 'relu' activation and 64 units. The last is also a fully connected layer with different units for each model and softmax activation, as all CNN models are multi-class classification models. The CNN models have been compiled using the optimizer 'adam', 'categorical_crossentropy' as loss and metrics as '["accuracy"]'.

Callbacks have been set for each model to stop them from overfitting. We have used Earlystoping and reduced learning rate to define the callbacks. Class weights have also been defined for each model to train them equally, even for classes with few images.

At least 15 epochs were applied to each model. For many CNNs, all epochs were not applied as training was stopped by callbacks to save the model from overfitting. Some of the models were also trained multiple times and with a higher number of epochs to increase their accuracy. The models were trained using transfer learning, where feature extraction was done on the training dataset, and validation was done using the test dataset.

A basic upload mechanism was also developed on Google Colab as well as DataSpell, where any image can be uploaded. This image is processed, classified, and the output is shown as an image with the disease name or healthy as the title. Figure 4 shows the output of an image provided to the CNN model.

Results and discussion

The proposed CNN structure could detect diseases in five fruits. The accuracy for each CNN model was above

Table 3. CNN architecture comparison table

Parameters	MobileNet	InceptionV3	DenseNet121	grape_leaves_classifier
Training accuracy (%)	63.35	81.00	97.23	99.75
Transfer accuracy (%)	93.95	96.79	98.02	98.64
No. of epochs	6	15	14	15
Training loss	0.69	0.38	0.07	0.03
Validation loss	0.40	0.12	0.05	0.02
Average time taken for each epoch (sec)	845	2575	2680	1410
Parameters	3,755,716	22,853,924	7,564,356	13,890,084

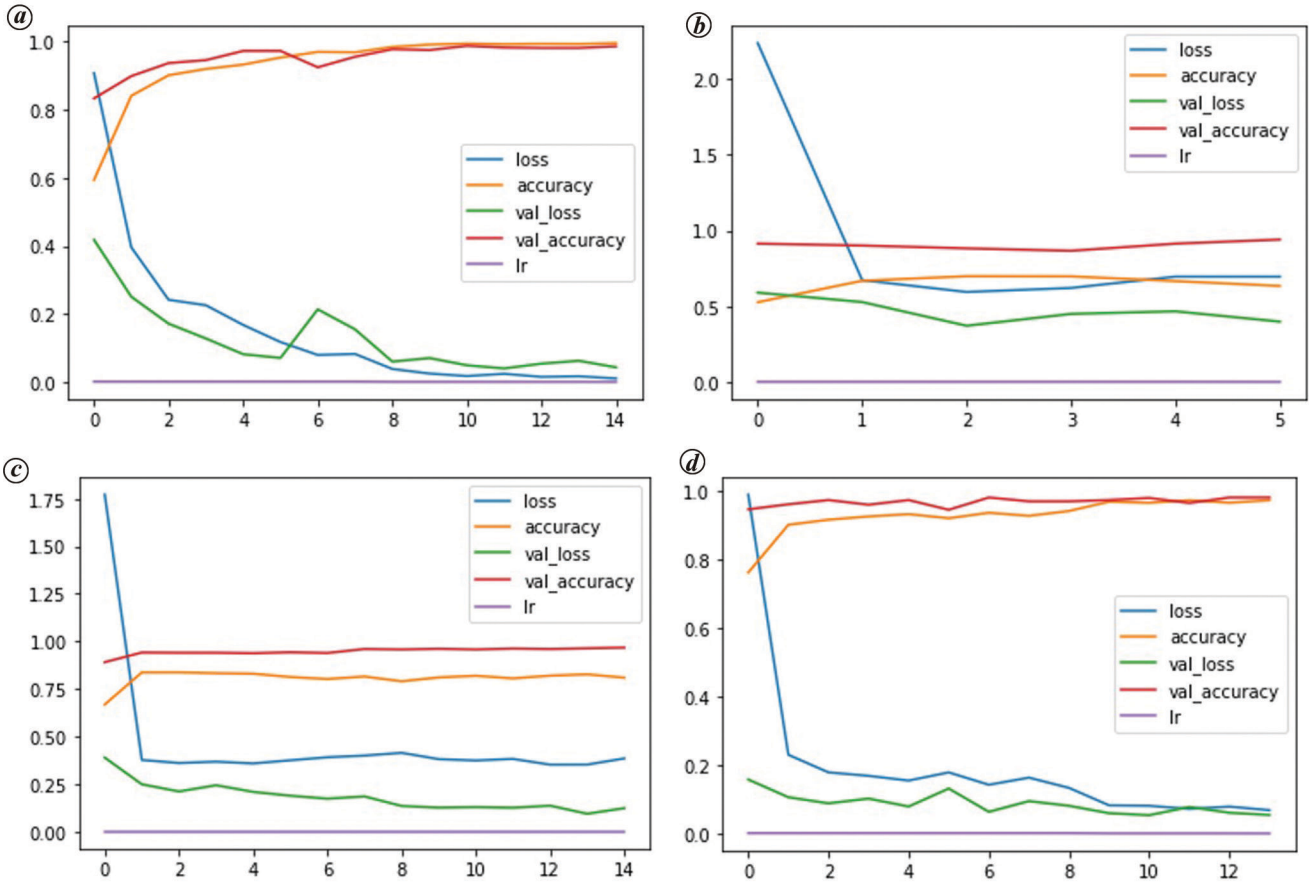

Figure 6. ***a***, The grape_leaves_classifier training data. ***b***, MobileNet training data. ***c***, InceptionV3 training data. ***d***, DenseNet121 training data.

90% for training or feature extraction and above 80% for the validation data. The loss score was also low for all the models. Figure 5 *a* and *b* show the training history plot and confusion matrix of the validation_classifier respectively.

The accuracy of the validation_classifier was 95.42% for the training data and 97.02% for the validation data. Loss for both training and validation data had decreased from 0.56 and 0.26 to 0.12 and 0.13 respectively, during the nine epochs while the learning rate remained constant.

All the other models were similarly trained and had achieved good accuracy. Table 2 shows the accuracy of all the models.

Comparison with pre-trained CNN architecture

We have compared the grape_leave_classifier model with three pre-trained CNN models, viz. MobileNet[22], DenseNet121 (ref. 23) and InceptionV3 (ref. 24). All the models were trained on the same dataset and the same number of epochs was applied (some models did not complete all the epochs as they were stopped by callbacks). Same callbacks were set while training all four models. Figure 6 *a–d* shows the training graphs of all the four models. Table 3 shows a comparison of all the models.

Table 3 shows that the proposed CNN model gives the best accuracy for training and testing datasets. Though

DenseNet121 shows similar accuracy, it uses a complex architecture which makes training hard and more time-consuming (it takes about 1270 sec more than the grape_leaves_classifier per epoch). While MobileNet takes less time than the other models, it gives the least accuracy. InceptionV3 has the largest number of parameters, but is unable to achieve higher accuracy.

Two other CNN architectures, i.e. ResNet50 and EfficientB2 were also trained on this dataset but were not included for comparison as their accuracy over epochs did not increase above 30%, even after multiple trials.

This comparison shows that the proposed CNN architecture is more suitable than other pre-trained architectures and provides better accuracy while also being simple, which allows it to be trained without extensive computation.

Conclusion and future scope

This study presents a combination of CNN models which can be used to classify five fruits into different diseased or healthy classes. The models have achieved an accuracy of above 90% for each of the five fruits and leaves.

In future, an android app-based model can be developed that can be used by anyone to identify fruit diseases. This would help farmers in rural areas save many of their fruit crops, as it would provide them with an easier and faster method to identify diseases and take actions to control their spread to other plants, or even save the current plants if the disease is detected in the early stages.

This study can be further continued, and more images be added to each dataset to achieve higher accuracy. More diseased classes could be added to the currently available fruits or more fruits can be added to the dataset. This study could be enhanced using classes with different stages of a single disease which can be identified and its stage informed to the user.

1. Gustavsson, J., Food and Agriculture Organization of the United Nations and ASME/Pacific Rim Technical Conference and Exhibition on Integration and Packaging of MEMS, N., Global food losses and food waste: extent, causes and prevention: study conducted for the International Congress 'Save Food!' at Interpack, Düsseldorf, Germany, 2011.
2. https://planthealthaction.org/news/plant-health-facts (accessed on 18 December 2021).
3. James, G. M. and Sujatha, S., Categorising apple fruit diseases employing hybrid neural clustering classifier. *Mater. Today: Proc.*, 2021.
4. Sembiring, A., Away, Y., Arnia, F. and Muharar, R., Development of concise convolutional neural network for tomato plant disease classification based on leaf images. *J. Phys.: Conf. Ser.*, IOP Publishing Ltd, 2021.
5. Xiao, J. R., Chung, P. C., Wu, H. Y., Phan, Q. H., Yeh, J. L. A. and Hou, M. T. K., Detection of strawberry diseases using a convolutional neural network. *Plants*, 2021, **10**, 1–14.
6. Ashwinkumar, S., Rajagopal, S., Manimaran, V. and Jegajothi, B., Automated plant leaf disease detection and classification using optimal MobileNet based convolutional neural networks. *Mater. Today: Proc.*, Elsevier Ltd, 2021, pp. 480–487.
7. Zia ur Rehman, Khan, M. A., Ahmed, F., Damaševičius, R., Naqvi, S. R., Nisar, W. and Javed, K., Recognizing apple leaf diseases using a novel parallel real-time processing framework based on MASK RCNN and transfer learning: an application for smart agriculture. *IET Image Process.*, 2021, **15**, 2157–2168.
8. Jogekar, R. N. and Tiwari, N., A review of deep learning techniques for identification and diagnosis of plant leaf disease. *Smart Innov., Syst. Technol.*, 2021, **182**, 435–441.
9. Wan Nurazwin Syazwani, R., Muhammad Asraf, H., Megat Syahirul Amin, M. A. and Nur Dalila, K. A., Automated image identification, detection and fruit counting of top-view pineapple crown using machine learning. *Alexandria Eng. J.*, 2022, **61**, 1265–1276.
10. Verma, S., Chug, A. and Singh, A. P., Application of convolutional neural networks for evaluation of disease severity in tomato plant. *J. Discrete Math. Sci. Cryptogra.*, 2020, **23**, 273–282.
11. Alim, M. M. F., Subiyanto and Sartini, Identification of diseases in tomato leaves using convolutional neural network and transfer learning method. *J. Phys.: Conf. Ser.*, 2021, **1918**.
12. Ilic, M., Ilic, S., Jovic, S. and Panic, S., Early cherry fruit pathogen disease detection based on data mining prediction. *Comput. Electron. Agric.*, 2018, **150**, 418–425.
13. Anton, A., Rustad, S., Shidik, G. F. and Syukur, A., Classification of tomato plant diseases through leaf using gray-level co-occurrence matrix and color moment with convolutional neural network methods. *Smart Innov., Syst. Technol.*, 2021, **182**, 291–299.
14. Ashok, V. and Vinod, D. S., A novel fusion of deep learning and android application for real-time mango fruits disease detection. *Adv. Intell. Syst. Comput.*, 2021, **1171**, 781–791.
15. Brahimi, M., Boukhalfa, K. and Moussaoui, A., Deep learning for tomato diseases: classification and symptoms visualization. *Appl. Artif. Intell.*, 2017, **31**, 299–315.
16. Abadi, M. *et al.*, TensorFlow: large-scale machine learning on heterogeneous distributed systems, 2016.
17. Chollet, F. and others, Keras, 2015; https://github.com/fchollet/keras
18. Buitinck, L. *et al.*, API design for machine learning software: experiences from the Scikit-learn project, 2013.
19. Mckinney, W., Data structures for statistical computing in Python, 2010.
20. Harris, C. R. *et al.*, Array programming with NumPy. *Nature*, 2020, **585**, 357–362.
21. Hunter, J. D., Matplotlib: a 2D graphics environment. *Comput. Sci. Eng.*, 2007, **9**, 90–95.
22. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T. and Andreetto, M., MobileNets: efficient convolutional neural networks for mobile vision applications, 2017.
23. Huang, G., Liu, Z., van der Maaten, L. and Weinberger, K. Q., Densely connected convolutional networks, 2017.
24. Szegedy, C., Vanhoucke, V., Ioffe, S. and Shlens, J., Rethinking the inception architecture for computer vision, 2015.

Received 13 February 2022; accepted 20 April 2022

doi: 10.18520/cs/v122/i11/1315-1320