International Journal of Modern Agriculture, Volume 10, No.1, 2021
ISSN: 2305-7246

Deep Learning Models for Beans Crop Diseases: Classification and Visualization Techniques

Priyanka Sahu[1], Anuradha Chug[2], Amit Prakash Singh[3], Dinesh Singh[4], and Ravinder Pal Singh[5]

[1,2,3]University School of Information, Communication, and Technology
Guru Gobind Singh Indraprastha University, New Delhi, India
[4,5]Division of Plant Pathology, Indian Agricultural Research Institute, New Delhi, India
[1]Er.priyankasahu40@gmail.com, [2]Anuradha@ipu.ac.in, [3]Amit@ipu.ac.in, [4]Dinesh_iari@rediffmail.com, and
[5]Ravinder.20033@gmail.com

Abstract- Crop diseases highly inhibit their growth. It may cause a critical loss of yield in crops; thus, respective crop quality or quantity gets affected. This is the reason why the detection of the disease in crops plays a significant role in the field of agriculture. Detection of crop diseases using some automatic techniques is helpful as it minimizes a massive work of supervision in big fields of production. It identifies the early symptoms of diseases in crops, i.e., as when they start to become visible on the plant leaves. In this study, beans crop leaf images were used in training for the classification, with a total of 1296 leaf images. Two Deep Learning models, namely, GoogleNet and VGG16 have been used to automatically extract the features from the images fed to the trained network. For training, bean crop leaves were classified into three different categories (classes), namely, Angular Leaf Spot, Beans Rust, and Healthy. Experimental results show that GoogleNet performs better than VGG16 with an accuracy of 95.31%. Visualization approaches, namely, Visualization of Intermediate layer activations, Visualization of the CNN filter, and Visualization of Heat Maps were used for analyzing, understanding the symptoms, and localization of diseased regions in the leaves. Moreover, it helps the naïve users to understand how a convolutional neural network works internally "instead of a black box" to identify and classify the diseased regions in an image.

Keywords- *Deep learning, classification, visualization, activation map, CNN.*

I. INTRODUCTION

Deep Learning (DL) has been started in 1943, as a new subcategory of Machine Learning(ML) when threshold logic was proposed to form a learning (computer) model that resembles the brain of humans. The evolution of research in this field can be categorized into 2-time frames: starting from 1943 to 2006 and from 2012 to the present. In its initial phase of developments, Backpropagation [1], Hand-written text recognition [2], chain-rule [3], and training problems were seen [4,5]. Subsequently, there were a lot of architectures/techniques that were proposed for multiple applications like the healthcare sector [6], marketing [7], image recognition [8–13], and text recognition [2,14,15]. Among all the frameworks, AlexNet [16] is observed as a benchmark in the area of DL, after winning the ImageNet challenge (ILSVRC) in 2012. After this, various architectures were proposed to overcome the research gaps seen previously. Several well-known performance metrics such as training/validation accuracy and loss [17,18], top-5%/top-1% error [8,10,16,19], classification accuracy (CA) [20–22], F1-score [23,24], precision and recall [9,17,23] were used to evaluate the results of these architectures.

As DL frameworks began to make advancements with the time, they were deployed in the field of image classification and recognition. These frameworks have also been introduced in various agricultural applications, e.g., plant leaves classification was carried out by deploying author- modified Convolutional Neural Network (CNN) with random forest (RF) classifier. Among 32 crop species, its performance was observed using CA at 97.3% [25]. In studies [26,27] and [28], authors performed implementations for fruit and leaf counting. For the classification of different crop types, Kussul et al. [29] implemented a user-modified CNN, Mortensen, et al. [18] applied VGG16, Rubwurm et al. [17] proposed LSTM, and Rebetez et al. [30] deployed CNN with RGB histogram. In this paper, a performance comparison has been made between the two pre-trained models GoogleNet and VGG16, in order to classify the healthy and diseased leaves of bean crops.

The rest of the paper is framed into the following sections. Section 2, gives some of the insights of DL. Section 3, discusses the Materials used for crop disease detection. Section 4, describes the experimental analysis using pre-trained models deployed over a small data sample. Section 5, Visualizing the learning process on CNN. Finally, section 6 concludes the study.

II. INSIGHTS OF DL

A. Applicability of DL for crop disease detection

Many DL architectures/models were developed soon after the famous AlexNet [16] for image segmentation, identification, and classification. This section shows some of the researches carried out using well-known DL models for the detection and classification of crops' diseases. In most of the studies, the PlantVillage dataset has been commonly used as it comprises 54,306 images of 14 distinct crops with 26 crop diseases [9]. LeNet was implemented to identify the diseases in banana leaves. F1-score and CA were applied to evaluate the model's performance in Gray Scale and Color modes [23]. In the study [31], the author evaluated a modified version of LeNet architecture that was deployed to identify olive crop diseases. Image segmentation technique along with edge maps was applied to spot the crop diseases. The same model was implemented in the study [32] to identify and classify the diseases in soybean crops. In order to detect vine crop diseases in UAV images, Kerkech et al. [33] combined the color space and vegetation indices with the LeNet model. Zhang et al. [34] have implemented the three CNN frameworks; AlexNet, ResNet, and GoogLeNet to identify the diseases in tomato leaves. Training and validation accuracy was computed to measure the performance of the architectures; ResNet gave the best results among all.

B. Data sources

It was observed that mainly large image datasets were applied over the DL architectures. In some cases, datasets comprised of thousands of images, either real images [9,20] or processed by the author [18,28]. Many datasets originated from publicly-available datasets, e.g. LifeCLEF, Flavia, PlantVillage, UC Merced, and Malayakew. Moreover, several other datasets compromised of the real images captured by the researcher according to their needs [11,22,37,38]. These images were taken either by UAV [4,30,39], airborne [40], satellite-based remote sensing [17,29], or using fields sensors [41]. In general, data requirement increases with the complexity of the problem, e.g., more training data is needed in DL when there is a small variation in between classes and the need is to identify a large number of classes in the dataset [9,17,22].

C. Data- pre-processing

It comprises the pre-processing of the provided data into floating-point vectors, the data readable by a CNN. The major part of related work done includes certain image pre-processing steps that were performed on the images prior to the training or their extracted features applied at the input layer of the DL architecture. Some well- known pre-processing techniques were image resizing (resized to 60x60, 96x96, 128x128, 256x256 pixels), data annotations [37,42], and image segmentation (used to highlight the regions of interest [9,11,20,43,44] and to increase the dataset size [30,45]). Some pre-processing techniques were also deployed for noise removal from images such as background removal [9,21], non-green pixel removal [20], extraction of foreground pixels [46]. Other techniques involved bounding boxes formation [21,42], conversion of image dataset to grayscale [23], or HSV color model [46]. Furthermore, in some studies, the features extracted from the images were fed to the input terminal of the DL model such as statistical and shape features [25], wavelet transformations [47], histograms [22,25,30], GLCM features [48], and PCA filters [22].

D. Data augmentation

Various data augmentation techniques [16] have been applied in the literature to enhance the diversity of image data for training models without adding new data. It helps to increase the overall learning process and the efficiency of the model. Augmentation procedure is especially significant for small datasets [11,18,38,49] for the training of DL models, as it helps in the generalization of data through serving the model with a variety of data. The use of data augmentation was also observed in the researches, where DL models were trained using synthetic images and were validated/tested using the real images [18,28].

E. Performance metrics

Various performance metrics have been used by the researchers to evaluate the performance of the model, each being precise to the DL model deployed in the study. In Table 1, these metrics are defined along with their used symbol. In some studies where the term accuracy is used without defining its meaning, we considered it as classification accuracy (CA). It has been deployed as the most commonly used metric. F1- score, RMSE, IoU, RFC are some other popular performance metrics. It was observed that some papers deployed a combination of metrics for the prediction of the model [50].

Table 1. Performance metrics deployed in studies under review.

Performance Metric	**Definition**	**Symbol**	**Reference**
Classification Accuracy	It is the % correct prediction from the total ones. CA= $\frac{TP+TN}{(TP+TN+FP+FN}$ Notations, TP= true positive TN= true negative FP= false positive FN= false negative	CA	[25,40,43,46,51] [20–22]
Precision	It is a fraction of the correct prediction from the total relevant results. P = $\frac{TP}{TP+FP}$	P	[9,17,23,50]
Recall	It is a fraction of True Positive from the total number of True Positive and false negatives. R = $\frac{TP}{TP+FN}$	R	[9,17,23,50]
F1-score	Defined as the harmonic mean of precision and recall. F1 = $\frac{2*TP*FP}{TP+FP}$	F	[23,24]
Mean Square Error	Mean of the square of the errors between predicted and observed values.	MSE	-
Root Mean Square Error	Standard deviation of the differences between predicted values and observed values.	RMSE	[41,52]
Ratio of total fruits counted	It was computed as the ratio of the predicted count value (of fruits), and the actual count. The actual count was calculated by taking the average of the model.	RFC	[28,42]
Intersection over Union	A metric that evaluates predicted bounding boxes, by dividing the area of overlap between the predicted and the ground-truth boxes, by the area of their union.	IoU	[18,50]

III. MATERIALS AND METHODS

In order to perform the implementation of DL architectures, various steps are needed; begin from the dataset collection to performance analysis and visualization mappings, the complete procedure is shown in figure 1. Initially, the input data is collected [9] and then split into two portions, 80-20 ratio of training and validation set. Soon after, DL architectures are deployed over the dataset with pre-training and without pre-training, and training/validation curves are drawn to represent the significance of the architectures. Moreover, performance metrics are applied to the classification of images (crop disease). Various visualization techniques are also mapped on the test data in prediction mode.

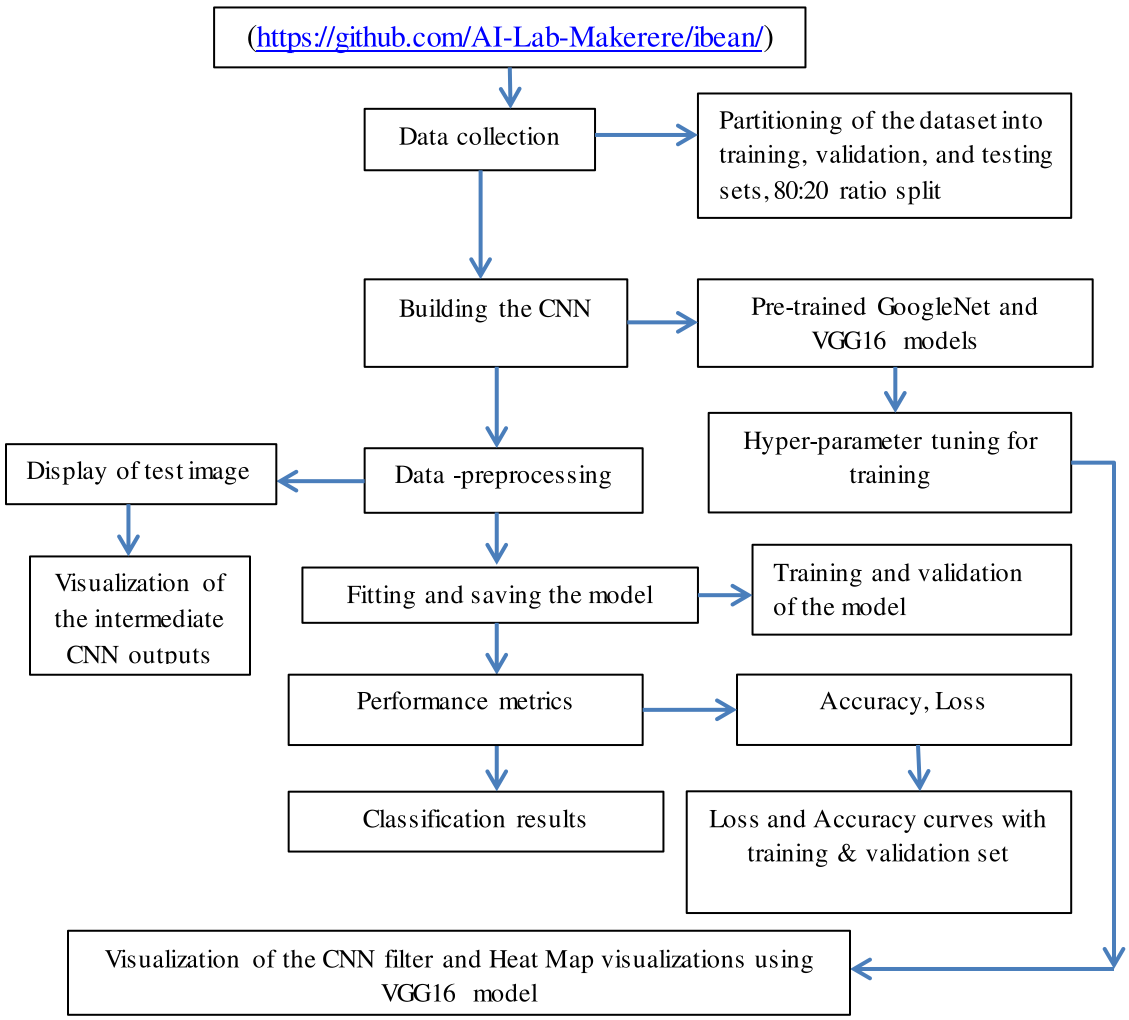


Figure 1. Shows the beans crop classification methodology using CNN.

A. Pre-trained models

For classification of crop diseases, DL models, especially CNN's, are trained directly over raw input images. Consequently, the DL models result in learning of the extracted features from input images without the involvement of any kind of manual help (human-intervention). In other words, automatic feature extraction occurs along with the training of the classifier. We have used two CNN models, namely, GoogleNet and VGG16. These frameworks were presented in computer vision challenges such as ImageNet and got some winning positions. The motive is to deploy these models for the identification of crop diseases.

1) VGG16

VGG16 is a 16 layered CNN architecture with 3x3 convolutional filters deployed to enhance the depth of the network. It revealed substantial upgrading for the accuracy of image recognition over large scale. The weight configuration of VGG16 architecture is openly accessible. This model involved of 138 million parameters that mark it challenging to handle. To detect the diseases in wheat crops, Lu et al. [39] implemented two DL architectures, namely, VGG- FCN, and VGG- CNN. Furthermore, feature visualization was done for each block in these DL models. In another research [53], the VGG- CNN framework was implemented for identification of disease (Fusarium wilt) in radish in which K- means clustering algorithm was applied to detect the spots of diseases.

2) GoogLeNet

Szegedy et al. [19] have implemented a 22 layers deep CNN model for image detection and classification. The main significance of this model is to improve the utilization of the computational resources that were deployed in the network. With the constant computational budget, the width and depth of the CNN were increased in this model. Hebbian principle and the concept of multi-scale processing was used to optimize the quality of architecture. GoogLeNet gave a top-5 error rate of 6.67%, which is very similar to human-level performance.

B. Workstation specifications and deep learning framework

All the implementations were performed using GoogleColab (python 3) on a personal computer with GPU:

- Python 3.7,
- 1xTesla K80,
- 2496 CUDA cores, and
- 12GB GDDR5 VRAM.

Such kind of GPU specification is vital for reducing the learning time from days to a few hours. GPU support is very significant in the processing of ample examples in each iteration of learning. For the implementation of a DL, there is a need for committed software and hardware to speed up the training.

C. Dataset

The dataset was choosen from the GitHub (https://github.com/AI-Lab-Makerere/ibean/). It comprised of the beans crop leaf images taken from the real field using a smartphone. Samples of the leaf images according to the divided classes are shown in figure 2. Table 2 show a description of the used dataset. This dataset holds 1296 images split into three classes. We have used three categories (labels) for the identification of diseases in crops.

Table 2. Summarization of Bean dataset.

Name of disease	Total number of image
Bean Rust	436
Angular Leaf Spot	432
Healthy	428
Total	1296

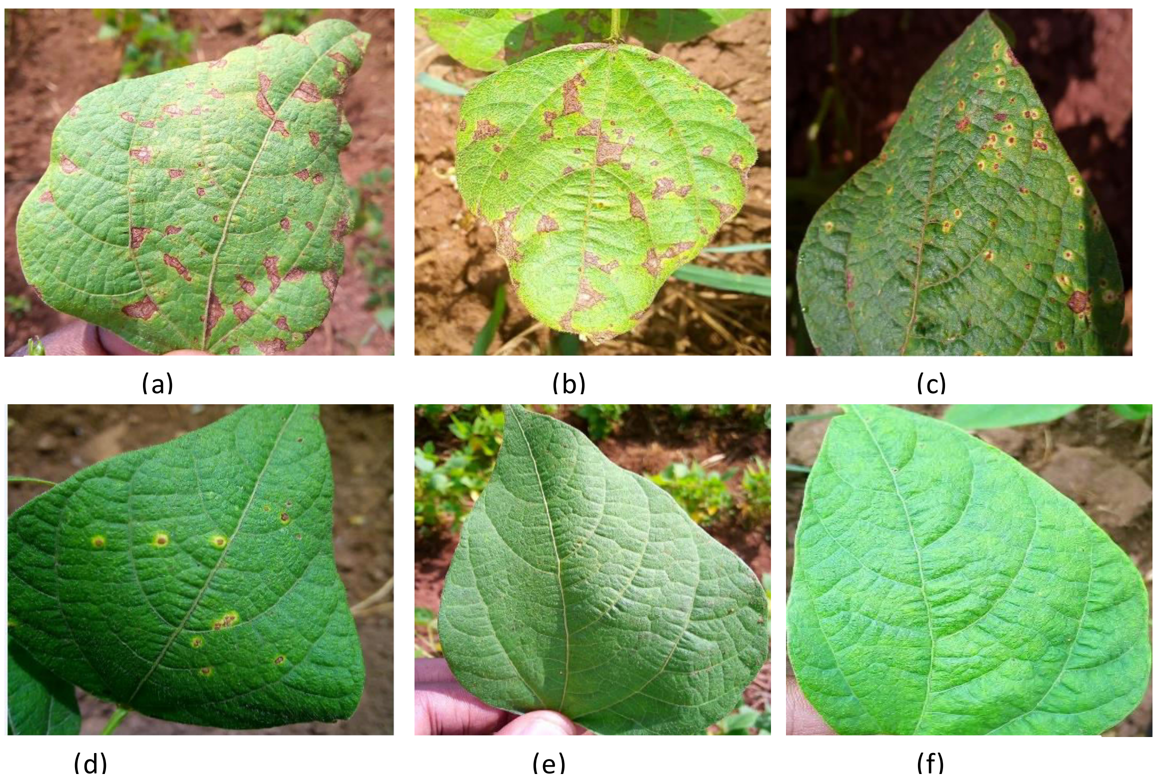


Figure 2. shows the beans crop images as follows: (a), (b) represents the Angular Leaf Spot, (c), (d) represents the bean rust disease, and (e), (f) represents the heathy leaf images.

IV. EXPERIMENTAL ANALYSIS

It is observed that the fine-tuning of pre-trained networks performed better than training from scratch (without pre-trained weights). Moreover, the fine-tuning of hyperparameters increases the accuracy of VGG16 from 0.896 to 0.9375, and GoogleNet from 0.901 to 0.9531. The impact of transfer learning is clarified by the capability of the network that reuses and transmit the features from one problem domain to another. These inherited features are used only with some minor changes in the last layers. Furthermore, the fine-tuning of hyperparameters is very helpful in situations where training datasets are small. The pre-trained models were trained over large datasets (ImageNet) with a higher number of labels, and these were reused over the smaller training examples. In addition, fine-tuning also benefits for training over the machines with a limited amount of memory in terms of GPU.

A comparison of the performance of pre-trained models is made with the models that were trained from scratch with randomly assigned network weights. It draws the effect of transfer learning on crop disease classification. Table 3 and figure 3 show the experimental results obtained with pre-trained and without pre-training.

Table 3. Experimentation results.

Deep architectures	Performance Measures	Without pre-training	With transfer learning
VGG16	Accuracy Loss	0.896 0.319	0.9375 0.2608
GoogleNet	Accuracy Loss	0.901 0.329	0.9531 0.2024

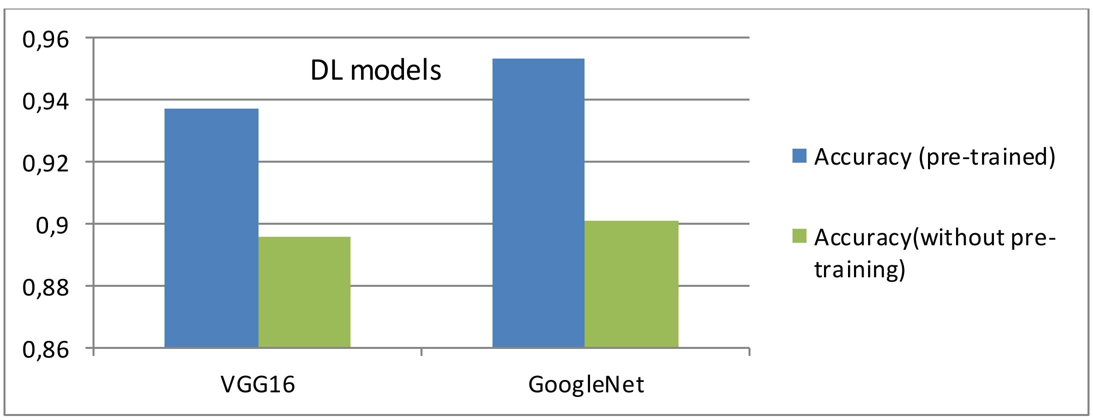


Figure 3. Analysis of DL models (with pre-trained weights versus training from scratch)

V. VISUALIZING THE LEARNING PROCESS IN CNN

It is observed that DL models are often categorized as "black box representations of learning," as these representations having difficulty in the extraction and presentation in a human-readable structure. However, this is not entirely true for CNNs, as CNN's represent the visual concepts of the convolutional layers. Here we present three visualization concepts of the CNN:

A. Visualization of the intermediate CNN outputs (Intermediate layer activations):
   It is useful to understand how the subsequent convolutional layer transfers their input from the first layer to the last one, and it also gives the idea of what CNN filters do.
B. Visualization of the CNN filter:

It is beneficial to understand precisely how visual patterns are receptive to a layer in a CNN.

C. Visualization of heat maps for class activation in an input (image):
It is beneficial to understand the parts of an input image that need to be identified to a particular class, or it allows a user for the localization of objects (regions of interest) in required images.

For the first approach (Activation visualization), We are using the small CNN, which is trained from scratch for beans disease. For the rest two methods, we have used the VGG16 framework.

A. Visualization of intermediate layer activation

Visualization of intermediate layer activations displays the feature maps, which are the resultant of the several convolutional and pooling layers in the CNN, provided a particular input. The output of the specific layer is termed as its activation [54,55]. It shows a view of how any input is segmented into distinct filters learned by the CNN. In this study, for feature maps visualization, three dimensions (channels), namely, height, width, and depth, are utilized. Each channel encodes its comparatively individual features.

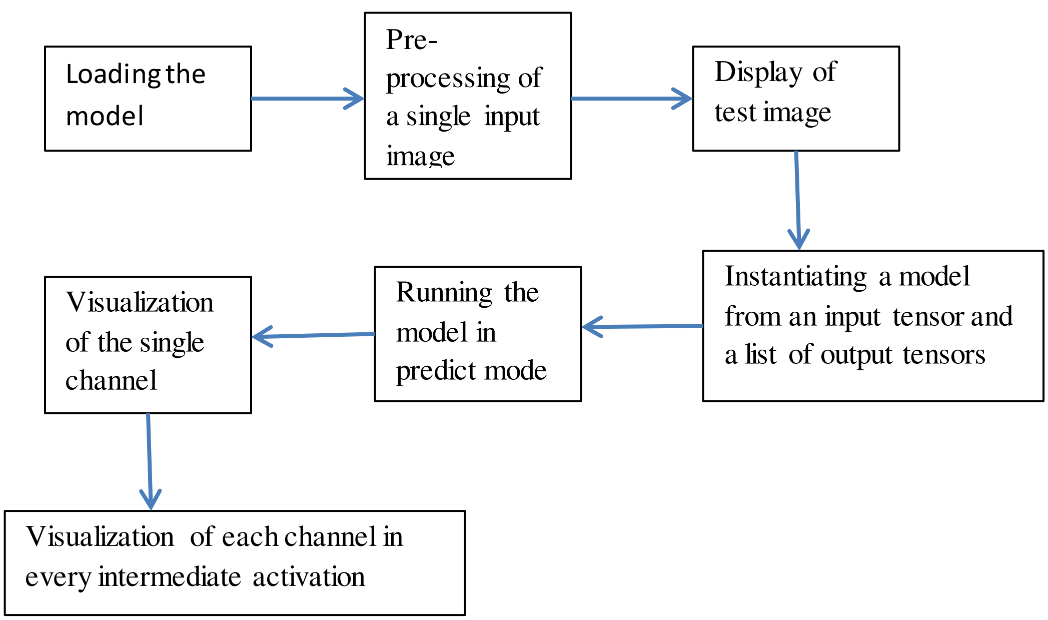
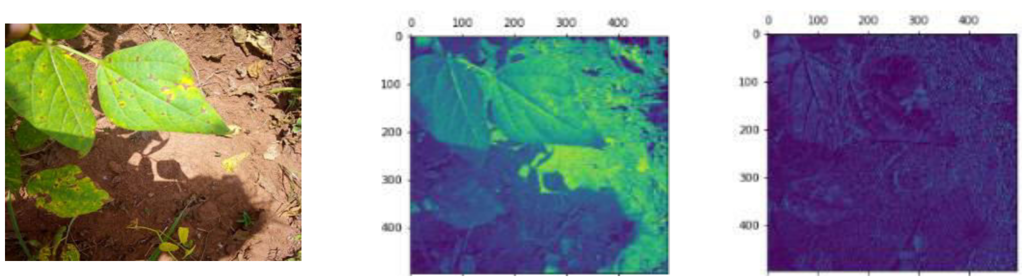


Figure 4. shows the flow of the visualizations of intermediate layer activation

The best way for visualization of such features is by individually plotting curves of the content of each channel in 2D- image format. Figure 4 shows the steps needed to proceed with the visualization of intermediate layer activations. The pre-processed image of a leaf (shown in figure 5) has 498*498 feature maps with one batch sample and 32 channels. It can be printed as: (1, 498, 498, 32). Feature map plotting for the 5^{th} channel and 12^{th} channel of the first layer activation is shown in figure 5. and the full activation visualization of the network is shown in figure 6. Every channel in the plotted map has eight activation maps of features. For extraction of the feature maps, a CNN model is needed that can carry the batches of input images and results in the outcomes of the activations for all convolutional and pooling layers. The model is realized using two parameters, namely, a list of input tensors and a list of output tensors. When an input image is fed to this model, it returns the layers' activation values. These are some characteristics of visualizations:

- There are several detectors such as edge detector, bright dot detector, luminance detector, etc. present in the first layer of the network. In this phase, the feature activation maps contain the complete information present in the provided image.
- As we go deeper, the feature activations will become more abstract and lesser visible for interpretation. Initial representations carry more visual information, and higher-level representations carry lesser visual information that is relevant to the classes of the image.
- The depth of the convolutional layer increases the sparseness of the feature activations that means, at the initial layer, input image activates all the maps (filters); however, in subsequent layers, many filters left as blank.

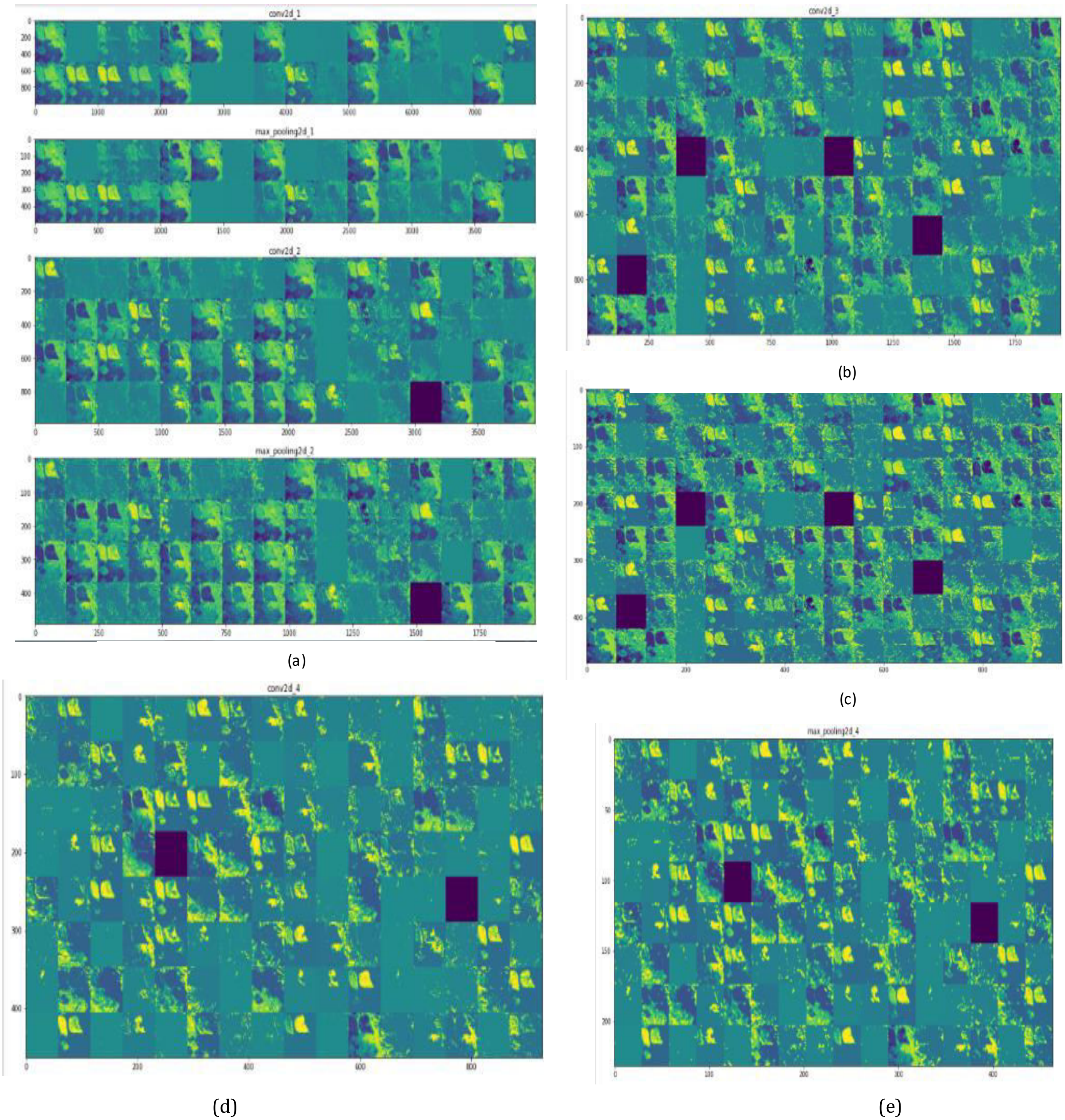


(a) Pre-processed image

(b) Visualization of the activations in the 5^{th} channel

(c) Visualization of the activations in the 12^{th} channel

Figure 5. Activation visualizations for channels.



(a), (b), (c), (d), and (e) are some intermediate layer activations of each channel.

Figure 6. Shows the intermediate channel activations

B. Visualization of CNN's filter

Filter visualization shows how the CNN layers are reflected in the world. Every layer in the CNN absorbs a pool of filters. The filters on CNN become progressively complicated and more advanced with the depth of the model. In the study [56], A feature visualization method was used to visualize the working of convolutional filters on the ImageNet dataset. Toda et al. [55] showed how CNNs diagnose crop diseases. It demonstrated the diagnosis of diseases for the plant's leaves taken from the PlantVillage dataset. The inspection of filters/ maps learned by the convolution network was used to show the visible patterns that are the response of an intended filter applied to the channel. Gradient descent was used in the input space for this functionality. Gradient descent was applied to the values in the input image of the CNN, which helps in maximizing the response of a particular filter (map). The resultant image is the one to which the selected filter is highly responsive. For the implementation of this approach, there is a need to form a loss function that helps in maximizing the value of a provided filter in a given convolutional layer [54].

Subsequently, the stochastic gradient descent was used for the adjustment of the values in an input image in order to maximize the feature map activation value. For the implementation of gradient descent, there was a need to find the gradient of the loss according to the input fed top of the model. Furthermore, gradient descent normalization was carried out to make the process smoother. It could be achieved by dividing the tensor through its square root of the average of the square of values in the tensor (L2 norm). This process ensured that the magnitude of the updates for the input image remains in the same range. Figure 7 shows the flow diagram for visualization of CNNs filter, and figure 8 displays an instance of the pattern for the 0th channel in layer block2_conv1.

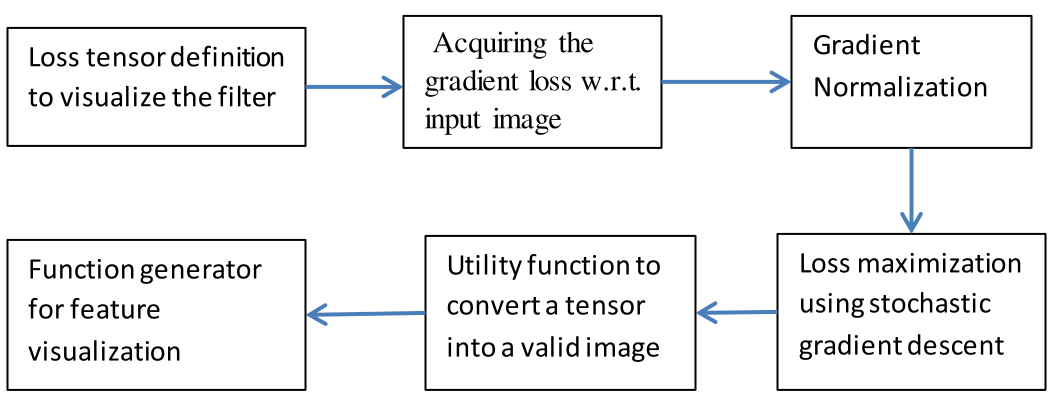


Figure 7. displays the flow diagram for visualization of CNNs filter.

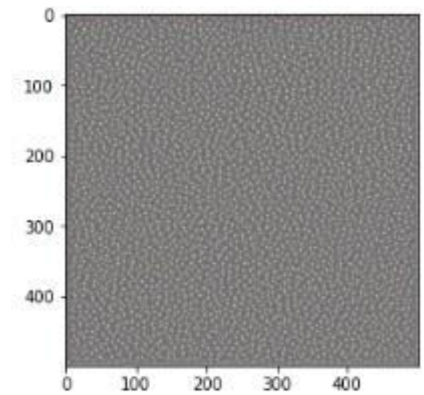


Figure 8. shows the visibility of pattern for the 0th channel in layer block2_conv1.

It was observed that filter '0' in layer block2_conv1 is receptive to a dot-like pattern (see figure 8). Similarly, visualization could be displayed for other layers also using the different available filters.

C. Heat maps visualization for class activation

Heat map visualizations are beneficial to understand which segment of an input image will be forwarded to a CNN for the final decision of classification. It also helps to debug the process of decision-making for a CNN, especially when there is any classification mistake. It also permits to show the location of particular objects in an input image. This visualization category is termed as class activation map (CAM) visualization and comprises the production of heat maps for class activation in the given image. A class activation heat map can be represented as a 2D grid of scores belonging to a particular output class, evaluated for each location over the input image in order to show the significance of every location for its respective class. For example, when input is fed into a CNN trained with images of plant diseases, CAM visualization permits for the generation of a heat map for class "disease" that indicates disease like spots present in an image. Fujita et al. [57] developed a plant diagnosis system for the severe viral manifestations in 9000 cucumber crop leaves images. They have deployed Heatmap visualizations to show the diagnostic regions in leaves images and captures significant features in their results.

"Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization" [58], a visualization approach given by Selvaraju et al. involves the convolution layer's output feature map, fed an input image, and weighing each channel in that feature map using the gradient of class w.r.t. the channel. This approach is illustrated in figure 9, using a pre-trained VGG16 network. Let us consider an input image of bean crop disease. The DL model trained with the image size dimensions of 500*500 pre-processed using some rules. After pre-processing, image sizes were adjusted according to the VGG16 architecture.

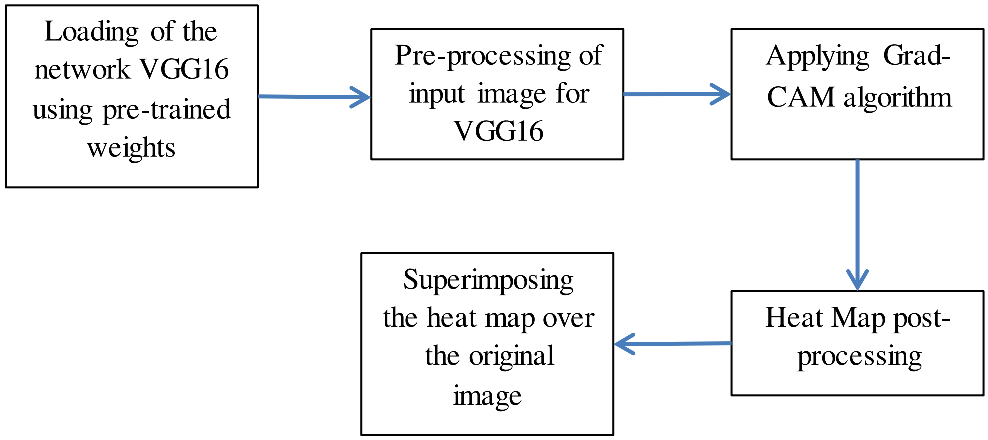
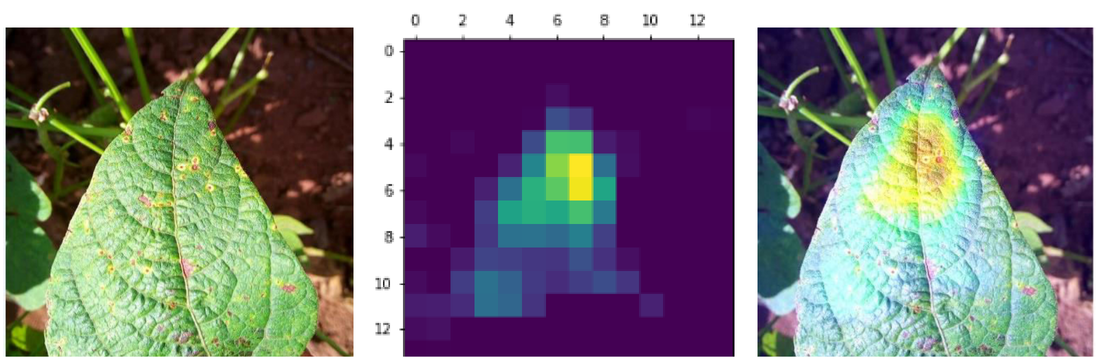


Figure 9. shows the steps needed for Heat maps visualization of class activation.

Grad- CAM algorithm [58] was applied for the visualization of the parts of the image that looks like the diseased spots present in the leaf of the bean crop. To accomplish the purpose of visualization, heat map normalization was done using heat map post-processing, and the normalization range was set up between 0-1. Figure 10. shows the effects of Heat Map class activation.



(a) Test image of rust- disease in bean crop leaf

(b) Leaf class activation Heat Map on the test image

(c) Superimposition of class activation Heat Map over the original test image

Figure 10. shows the effects of Heat Map class activation.

VI. CONCLUSION

In this paper, CNN based DL models are compared in order to carry the beans leaf disease (angular leaf spot and beans rust) classification. The experimental results show that GoogleNet performs better than VGG16 for disease classification. Furthermore, the experimentation also validates the use of pre-training (transfer learning) over the without pre-training (training from scratch). This study also performs some visualizations techniques, namely, Visualization of intermediate layer activation, Visualization of the CNN filter, and Heat maps based visualization for class activation. It visualizes the results of activation maps deployed in the intermediate convolutional layers and on the regions of the infected image. It helps the naïve users to understand the internal working of the network.

ACKNOWLEDGMENT

Authors are thankful to the Department of Science & Technology, Government of India, Delhi for funding a project on "Application of IoT in Agriculture Sector" through ICPS division. This work is a part of the project work.

REFERENCES

1. Ackley DH, Hinton GE, Sejnowski TJ. A learning algorithm for Boltzmann machines. Cogn Sci. Elsevier; 1985;9:147–69.

2. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE. Ieee; 1998;86:2278–324.

3. Dreyfus S. The numerical solution of variational problems. J Math Anal Appl. Academic Press; 1962;5:30–45.

4. Hinton GE, Osindero S, Teh Y-W. A fast learning algorithm for deep belief nets. Neural Comput. MIT Press; 2006;18:1527–54.

5. Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. Science (80-). American Association for the Advancement of Science; 2006;313:504–7.

6. Esteva A, Robicquet A, Ramsundar B, Kuleshov V, DePristo M, Chou K, Cui C, Corrado G, Thrun S, Dean J. A guide to deep learning in healthcare. Nat Med. Nature Publishing Group; 2019;25:24–9.

7. Siau K, Yang Y. Impact of artificial intelligence, robotics, and machine learning on sales and marketing. Twelve Annu Midwest Assoc Inf Syst Conf (MWAIS 2017). 2017. p. 18–9.

8. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. Proc IEEE Conf Comput Vis pattern Recognit. 2016. p. 770–8.

9. Mohanty SP, Hughes DP, Salathé M. Using deep learning for image-based plant disease detection. Front Plant Sci. Frontiers; 2016;7:1419.

10. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv Prepr arXiv14091556. 2014;

11. Sladojevic S, Arsenovic M, Anderla A, Culibrk D, Stefanovic D. Deep neural networks based recognition of plant diseases by leaf image classification. Comput Intell Neurosci. Hindawi; 2016;2016.

12. Wan J, Wang D, Hoi SCH, Wu P, Zhu J, Zhang Y, Li J. Deep learning for content-based image retrieval: A comprehensive study. Proc 22nd ACM Int Conf Multimed. 2014. p. 157–66.

13. Wu R, Yan S, Shan Y, Dang Q, Sun G. Deep image: Scaling up image recognition. arXiv Prepr arXiv150102876. 2015;7.

14. Goodfellow IJ, Bulatov Y, Ibarz J, Arnoud S, Shet V. Multi-digit number recognition from street view imagery using deep convolutional neural networks. arXiv Prepr arXiv13126082. 2013;

15. Jaderberg M, Simonyan K, Vedaldi A, Zisserman A. Deep structured output learning for unconstrained text recognition. arXiv Prepr arXiv14125903. 2014;

16. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Adv Neural Inf Process Syst. 2012. p. 1097–105.

17. Rußwurm M, Körner M. Multi-temporal land cover classification with long short-term memory neural networks. Int Arch Photogramm Remote Sens Spat Inf Sci. Copernicus GmbH; 2017;42:551.

18. Mortensen AK, Dyrmann M, Karstoft H, Jørgensen RN, Gislum R, others. Semantic segmentation of mixed crops using deep convolutional neural network. Proc Int Conf Agric Eng. 2016.

19. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, et al. Going deeper with convolutions. Proc IEEE Conf Comput Vis pattern Recognit. 2015. p. 1–9.

20. Dyrmann M, Karstoft H, Midtiby HS. Plant species classification using deep convolutional neural network. Biosyst Eng. Elsevier; 2016;151:72–80.

21. McCool C, Perez T, Upcroft B. Mixtures of lightweight deep convolutional neural networks: Applied to agricultural robotics. IEEE Robot Autom Lett. IEEE; 2017;2:1344–51.

22. Xinshao W, Cheng C. Weed seeds classification based on PCANet deep learning baseline. 2015 Asia-Pacific Signal Inf Process Assoc Annu Summit Conf. 2015. p. 408–15.

23. Amara J, Bouaziz B, Algergawy A, others. A Deep Learning-based Approach for Banana Leaf Diseases Classification. BTW. 2017. p. 79–88.

24. TÜRKO\uGLU M, Hanbay D. Plant disease and pest detection using deep learning-based features. Turkish J Electr Eng Comput Sci. The Scientific and Technological Research Council of Turkey; 2019;27:1636–51.

25. Hall D, McCool C, Dayoub F, Sunderhauf N, Upcroft B. Evaluation of features for leaf classification in challenging conditions. 2015 IEEE Winter Conf Appl Comput Vis. 2015. p. 797–804.

26. Itzhaky Y, Farjon G, Khoroshevsky F, Shpigler A, Bar-Hillel A. Leaf counting: Multiple scale regression and detection using deep CNNs. BMVC. 2018. p. 328.

27. Ubbens J, Cieslak M, Prusinkiewicz P, Stavness I. The use of plant models in deep learning: an application to leaf counting in rosette plants. Plant Methods. Springer; 2018;14:6.

28. Rahnemoonfar M, Sheppard C. Deep count: fruit counting based on deep simulated learning. Sensors. Multidisciplinary Digital Publishing Institute; 2017;17:905.

29. Kussul N, Lavreniuk M, Skakun S, Shelestov A. Deep learning classification of land cover and crop types using remote sensing data. IEEE Geosci Remote Sens Lett. IEEE; 2017;14:778–82.

30. Rebetez J, Satizábal HF, Mota M, Noll D, Büchi L, Wendling M, et al. Augmenting a convolutional neural network with local histograms-A case study in crop classification from high-resolution UAV imagery. ESANN. 2016.

31. Cruz AC, Luvisi A, De Bellis L, Ampatzidis Y. Vision-based plant disease detection system using transfer and deep learning. 2017 asabe Annu Int Meet. 2017. p. 1.

32. Wallelign S, Polceanu M, Buche C. Soybean plant disease identification using convolutional neural network. Thirty-First Int Flairs Conf. 2018.

33. Kerkech M, Hafiane A, Canals R. Deep leaning approach with colorimetric spaces and vegetation indices for vine diseases detection in UAV images. Comput Electron Agric. Elsevier; 2018;155:237–43.

34. Zhang K, Wu Q, Liu A, Meng X. Can deep learning identify tomato leaf disease? Adv Multimed. Hindawi; 2018;2018.

35. Ferentinos KP. Deep learning models for plant disease detection and diagnosis. Comput Electron Agric. Elsevier; 2018;145:311–8.

36. Fuentes A, Yoon S, Kim SC, Park DS. A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition. Sensors. Multidisciplinary Digital Publishing Institute; 2017;17:2022.

37. Bargoti S, Underwood J. Deep fruit detection in orchards. 2017 IEEE Int Conf Robot Autom. 2017. p. 3626–33.

38. Sørensen RA, Rasmussen J, Nielsen J, Jørgensen RN. Thistle detection using convolutional neural networks. 2017 EFITA WCCA Congr. 2017. p. 161.

39. Lu J, Hu J, Zhao G, Mei F, Zhang C. An in-field automatic wheat disease diagnosis system. Comput Electron Agric. Elsevier; 2017;142:369–79.

40. Chen Y, Lin Z, Zhao X, Wang G, Gu Y. Deep learning-based classification of hyperspectral data. IEEE J Sel Top Appl earth Obs Remote Sens. IEEE; 2014;7:2094–107.

41. Song X, Zhang G, Liu F, Li D, Zhao Y, Yang J. Modeling spatio-temporal distribution of soil moisture by deep learning-based cellular automata model. J Arid Land. Springer; 2016;8:734–48.

42. Chen SW, Shivakumar SS, Dcunha S, Das J, Okon E, Qu C, et al. Counting apples and oranges with deep learning: A data-driven approach. IEEE Robot Autom Lett. IEEE; 2017;2:781–8.

43. Grinblat GL, Uzal LC, Larese MG, Granitto PM. Deep learning for plant identification using vein morphological patterns. Comput Electron Agric. Elsevier; 2016;127:418–24.

44. Hashim N, Janius R Bin, Baranyai L, Rahman RA, Osman A, Zude M. Kinetic Model for Colour Changes in Bananas During the Appearance of Chilling Injury Symptoms. Food Bioprocess Technol. 2012;5:2952–63.

45. Yalcin H. Plant phenology recognition using deep learning: Deep-Pheno. 2017 6th Int Conf Agro-Geoinformatics. 2017. p. 1–5.

46. Lee SH, Chan CS, Wilkin P, Remagnino P. Deep-plant: Plant identification with convolutional neural networks. 2015 IEEE Int Conf image Process. 2015. p. 452–6.

47. Kamilaris A, Prenafeta-Boldú FX. Deep learning in agriculture: A survey. Comput Electron Agric. Elsevier; 2018;147:70–90.

48. Olaniyi EO, Adekunle AA, Odekuoye T, Khashman A. Automatic system for grading banana using GLCM texture feature extraction and neural network arbitrations. J Food Process Eng. Wiley Online Library; 2017;40:e12575.

49. DeChant C, Wiesner-Hanks T, Chen S, Stewart EL, Yosinski J, Gore MA, et al. Automated identification of northern leaf blight-infected maize plants from field imagery using deep learning. Phytopathology. Am Phytopath Society; 2017;107:1426–32.

50. Sa I, Ge Z, Dayoub F, Upcroft B, Perez T, McCool C. Deepfruits: A fruit detection system using deep neural networks. Sensors. Multidisciplinary Digital Publishing Institute; 2016;16:1222.

51. Pound MP, Atkinson JA, Townsend AJ, Wilson MH, Griffiths M, Jackson AS, et al. Deep machine learning provides state-of-the-art performance in image-based plant phenotyping. Gigascience. Oxford University Press; 2017;6:gix083.

52. Sehgal G, Gupta B, Paneri K, Singh K, Sharma G, Shroff G. Crop planning using stochastic visual optimization. 2017 IEEE Vis Data Sci. 2017. p. 47–51.

53. Ha JG, Moon H, Kwak JT, Hassan SI, Dang M, Lee ON, et al. Deep convolutional neural network for classifying Fusarium wilt of radish from unmanned aerial vehicles. J Appl Remote Sens. International Society for Optics and Photonics; 2017;11:42621.

54. Chollet F. Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek. MITP-Verlags GmbH & Co. KG; 2018.

55. Toda Y, Okura F, others. How convolutional neural networks diagnose plant disease. Plant Phenomics. AAAS; 2019;2019:9237136.

56. Erhan D, Bengio Y, Courville A, Vincent P. Visualizing higher-layer features of a deep network. Univ Montr. 2009;1341:1.

57. Fujita E, Uga H, Kagiwada S, Iyatomi H. A practical plant diagnosis system for field leaf images and feature visualization. Int J Eng Technol. 2018;7:49–54.

58. Selvaraju RR, Cogswell M, Das A, Vedantam R, Parikh D, Batra D. Grad-cam: Visual explanations from deep networks via gradient-based localization. Proc IEEE Int Conf Comput Vis. 2017. p. 618–26.