Chapter 12

Sugarcane leaf disease detection through deep learning

N.K. Hemalatha[a], R.N. Brunda[a], G.S. Prakruthi[a], B.V. Balaji Prabhu[b,c], Arpit Shukla[d], and Omkar Subbaram Jois Narasipura[c]

[a]*Dr. Ambedkar Institute of Technology, Bangalore, Karnataka, India,* [b]*Malnad College of Engineering, Hassan, Karnataka, India,* [c]*Indian Institute of Science, Bangalore, Karnataka, India,* [d]*National Institute of Technology Srinagar, Srinagar, Jammu and Kashmir, India*

1 Introduction

Agriculture contributes about 6.4% to the world's economy. Among the world's 226 countries, 9 countries hold agriculture as its dominant sector of economy contributor. China is the largest contributor, then comes India, and the United States takes third place. Among the commercial agricultural crops, sugarcane is one of the major commercial crops as it is used for the production of various by-products like sugar, fermented liquids like syrups and capsules, bagasse used for fuel, fiberboard and molasses used in distilleries for the manufacture of ethyl alcohol, butyl alcohol, citric acid, and so on.

Sugarcane is an ancient crop of the Austronesian and Papuan people. It was introduced to Polynesia, Island Melanesia, and Madagascar in prehistoric times via Austronesian sailors. It was also introduced to India and Southern China by Austronesian traders at around 1200–1000 BCE. It is the world's largest crop by production quantity, with 1.9 billion tons produced in 2015, with Brazil accounting for 41% of the world total. In 2012, the Food and Agriculture Organization estimated that it was cultivated on around 26 million hectares (64 million acres) in more than 90 countries. Sugarcane is a tropical, perennial grass that produces multiple stems by forming lateral shoots at the base, typically 3–4 m (10–13 ft) high and about 5 cm (2 in) in diameter. These stems grow into a cane stalk, which constitutes about 75% of the entire mature plant. A mature stalk is typically composed of 12%–16% soluble sugars, 2%–3% nonsugars, 11%–16% fiber, and 63%–73% water. The sugarcane crop is sensitive to climate, soil type, irrigation, disease control, fertilizers, insects, varieties, and the harvest period. The average yield of cane stalk is around 60–70 tons per hectare (24–28 long ton/acre; 27–31 short ton/acre) per year. However, this

Deep Learning for Sustainable Agriculture. https://doi.org/10.1016/B978-0-323-85214-2.00003-3

figure can vary between 30 and 180 tons per hectare depending on the knowledge and crop management approach used in sugarcane cultivation. Sugarcane is a cash crop but is also used as livestock fodder.

Though sugarcane is spread vast across the world, it is a painstaking job for farmers all over when the crops are subjected to unforeseeable climatic fluctuations and menacing pests and diseases. Production quantities are affected by pests and diseases that occur in sugarcane plants. Immense commercialization in the agricultural field is also a contributing factor for plant diseases. There are multifarious diseases that affect the crop in yield and quality. Some of these are detected by farmers when they visually inspect the leaves. However, most of the diseases go undetected, leading to huge losses to farmers. Therefore, it is crucial to identify the type of infestation to aid in controlling its damage. To prevent these diseases, costly methods and various pesticides are used in the agriculture. The widespread use of these chemical methods harm plant health and human health and affects the environment negatively. Also, these methods, increases production costs. In such cases, farmers are suggested for precise agricultural applications, where medicines are sprayed only to the affected area. It is necessary to determine the regions where the plant diseases occur and spread. Deep learning (DL) is widely used in precision agriculture. It is helpful for image-based plant disease recognition. DL is the state-of-the-art machine learning (ML) method that utilizes artificial neural networks (ANNs) with hidden layers.

In the literature, various researchers have worked on detection of plant diseases to help farmers in identifying diseases at an early stage and take necessary actions to deal with the same.

Sladojevic et al. (2016) developed a leaf disease detection (LDD) system to automatically classify and detect 13 different classes of plant diseases from leaf images using a deep convolution neural network approach consisting of five convolution and three fully connected layers. Even though the system achieves a good performance, improper implementation of the system for use by farmers, like a globally hosted website, makes it difficult to for farmers. Maniyath et al. (2018) developed a LDD system using a ML algorithm "random forest" in order to differentiate between healthy and diseased. But the major drawback in this implementation is that they used a ML approach instead of DL, which gives more efficient training results by effective feature learning. Kulkarni (2018) proposed a crop disease detection system that included 38 different classes of diseases and the image dataset consisted of 54,306 images. The detecting model was prepared using the pretrained model named MobileNet. There is no proper description of the crops and diseases Kulkarni considered for training the model, and the implementation of the system as a smartphone application is one among his future enhancements. Toda and Okura (2019) proposed a convolution neural network-based (CNN-based) approach to diagnose plant diseases. They considered 38 varieties of plants, classifying them into two classes, namely, diseased and healthy. They used an InceptionV3 pretrained model for training the dataset and visualized the results based on output,

features, semantic dictionary, and Padilla et al. (2019) came up with the main objectives of producing a system capable of identifying a yellow spot diseased leaf among sugarcane leaves using a support vector machine algorithm. The system was designed with a Noir camera integrated with a Raspberry Pi and an LCD monitor that helped for observation and classification. Saleem, Potgieter, and Arif (2019) developed DL architectures along with visualization techniques for plant disease classification. They implemented the dataset on various algorithms from LeNet until the latest VGG inception. Mohanty et al. (2016) developed a model using a DL approach using a public image dataset of 54,306 images of diseased and healthy plant leaves under controlled conditions. They trained for 14 varieties and 26 classes of diseases. They achieved an accuracy of 99.35%. Patil and Pawar (2017) developed an Android application that identifies a plant species based on a photo captured through a mobile camera. The morphological features of the leaves that computes the angle code histogram, and then the classification is done based on a novel combination of the computed metrices.

The literature reveals that works carried by various researchers focused on developing an improved accurate model. But the works do not reach farmers to take the advantage of the same. So, it is very important to develop a system that can easily accessed and used by any farmer. To address this issue, we propose a DL neural network architecture in which the type of disease afflicting the sugarcane crop is predicted by training the model on images of affected leaves. The advancements in telecommunication and its technology made smartphone affordable. So, the proposed work majorly involves the smartphone as an end device to made it possible for the research to reach the farming community.

This chapter discusses the methodologies used, which includes dataset, proposed system architecture, and the proposed Android application. Then there is a discussion about the experimentation conducted, the results are discussed, and conclusions are presented.

2 Methodology

This section discusses the dataset, system architecture and network model used and also the proposed SAFAL-FASAL Android application.

2.1 Dataset

The quality, relevancy, and availability of the data directly affects the goals of DL model. Incomplete or inaccurate data sets will train the model like an illiterate human who cannot understand the environment better. Hence, choosing the right data for the model will also help achieve accurate results. Therefore, a model deserves the best data that are precisely labeled, which can only help the model achieve the best level of accuracy at an affordable cost.

In this work, we have used a dataset containing 2940 images of sugarcane leaves belonging to 6 different classes (5 diseases and 1 healthy), as shown in Fig. 1. All these images were taken in natural environments with numerous variations. The images were taken at various cultivation fields, including the University of Agricultural Sciences and nearby farms in Bangalore and Mandya. All the images were taken using phone cameras at various angles, orientations, and backgrounds accounting for most of the variations that can appear for images taken in the real world. The dataset was collected with the company of experienced pathologists. The distribution of the images into different classes is detailed in Table 1.

2.2 Leaf disease detection system architecture

Fig. 2 shows the architecture of the proposed LDD system for detection of sugarcane leaf diseases. The system architecture brief about the sequence of

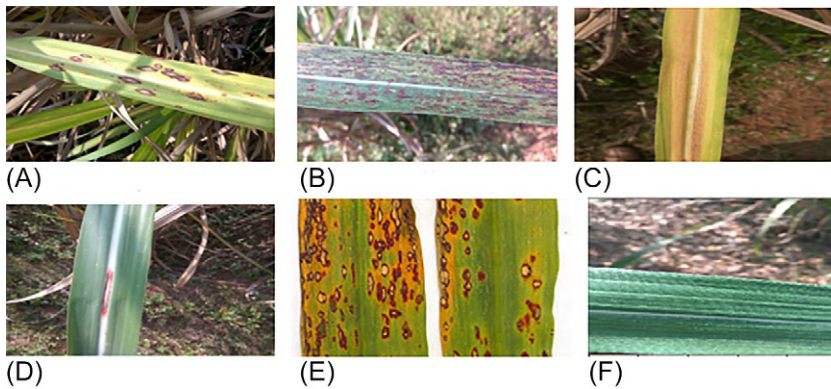


FIG. 1 Sugarcane leaf disease classes along with healthy class, (A) Helminthospora, (B) rust, (C) yellow leaf, (D) red rot, (E) Cercospora leaf spot, and (F) healthy leaf.

TABLE 1 Distribution of images into different classes.

Sl. No	Class	Count
1	Cercospora Leaf Spot	346
2	Helmanthospura Leaf Spot	410
3	Rust	382
4	Red Rot	454
5	Yellow Leaf Disease	420
6	Healthy	928

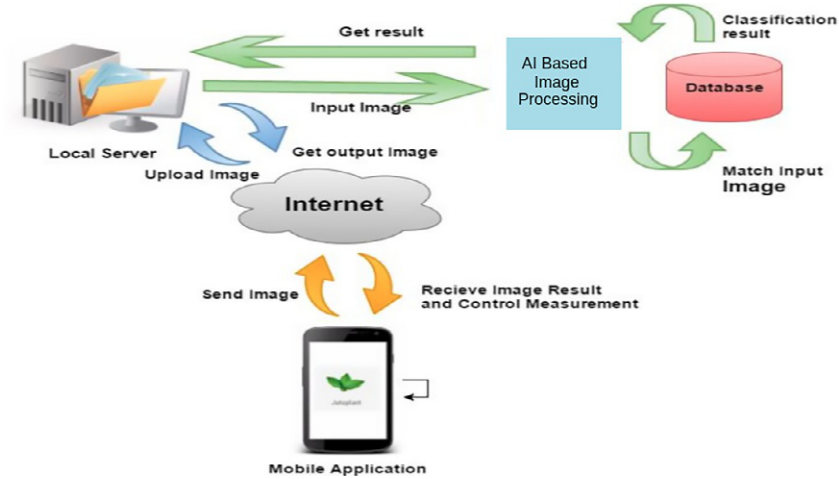


FIG. 2 LDD system architecture.

operations need to be followed in detecting the leaf disease. The SAFAL-FASAL mobile application is developed to provide information to farmers about the diseases affecting their crop so that they can take necessary measures to avoid losses. Any user with the SAFAL-FASAL app installed on their mobile can capture images of the sugarcane leaf through the camera or by selecting an existing leaf image from the phone's gallery and upload the image to detect the type of disease affecting the sugarcane plant. Upon uploading the leaf image through the mobile, SAFAL-FASAL will send the image to the server for analysis. At the server end, the image is processed by the developed LDD model to detect the corresponding disease. The LDD model predicts the class of the disease infected by that sugarcane plant. The predicted class is displayed on the mobile screen. The user can either view the class of the disease affecting the leaf or view that the leaf is healthy.

2.3 Leaf disease detection model architecture

CNNs are an evolution of traditional ANNs, focused mainly on applications with repeating patterns in different areas of the modeling space, especially image recognition. For image recognition applications, several baseline architectures of CNNs have been developed, which have been successfully applied to complicated tasks of visual imagery. Some of the well-known CNN architectures used for the task of classification are LeNet-5, AlexNet, VGG-16, Inception-v1, Inception-v3, ResNet-50, Xception, Inception-v4, Inception-ResNets, and ResNeXt-50. The studies prove that LeNet-5 outperforms other network models in the task of image classification. So, LeNet-5 architecture is used as a network

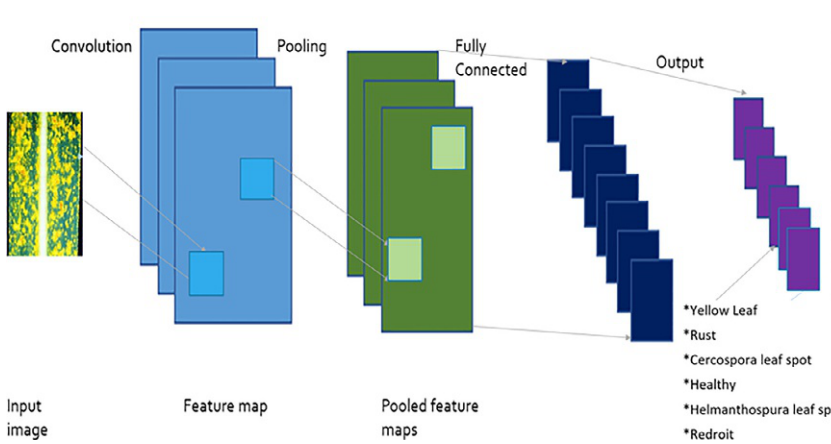


FIG. 3 LDD model network architecture.

architecture to develop LDD model. Fig. 3 shows the network architecture of LDD model, which uses LeNet-5 as a base model.

The proposed LDD model architecture consists of nine layers, including three convolutional layers (C1, C3, and C5), two subsampling (pooling) layers (S2 and S4), and two fully connected layer (F6), followed by the output layer. The input layer is built to take in $255 \times 255 \times 3$, and these are the dimensions of images that are passed into the next layer.

The official first layer, convolutional layer C1, produces as output 32 feature maps and has a kernel size of 5×5. The kernel/filter is the name given to the window that contains the weight values that are utilized during the convolution of the weight values with the input values.

A subsampling (pooling) layer S2 follows the C1 layer. The S2 layer halves the dimension of the feature maps it receives from the previous layer to $127 \times 127 \times 32$ with the filter size 2×2; this is known commonly as down-sampling. This is to decrease the computational power required to process data through dimensionality reduction. Furthermore, it is useful for extracting the dominant features that are rotational and positional invariant. The S2 layer also produces 32 feature maps, each one corresponding to the feature maps passed as input from the previous layer. There are two types of pooling:

(1) Max pooling returns the maximum value from the portion of the image covered by the kernel.
(2) Average pooling returns the average of all the values from the portion of the image covered by the kernel.

Pooling is followed by the convolutional layer C3 with 64 filters applied on the feature maps, which are the output of the previous layer S2. The fourth layer (S4) is again a pooling layer with filter size 2×2. This layer is the same as

the second layer (S2) except it has 64 feature maps so the output will be reduced to 63x63x64. The fifth layer (C5) is a fully connected convolutional layer for output mapping and for producing the predictions.

The sixth layer is the dense layer, which had a specified number of units or neurons within each layer, while the output layer has six units.

The last layer has six units that correspond to the number of classes that are used in the dataset. The activation function for the output layer is a softmax activation function.

Softmax is an activation function that is utilized to derive the probability distribution of a set of numbers within an input vector. The output of a softmax activation function is a vector in which its set of values represents the probability of an occurrence of a class/event. These values within the vector all add up to 1.

2.4 SAFAL-FASAL android application

The system design for the proposed SAFAL-FASAL application is given in Fig. 4. The user has to capture the image of the sugarcane leaf and upload it. This image is fed as input to the trained TensorFlow Lite (TFLite) model for processing; after processing, the model outputs the predicted class. The predicted disease will then be displayed in the application for the user to learn about the disease affecting the crop.

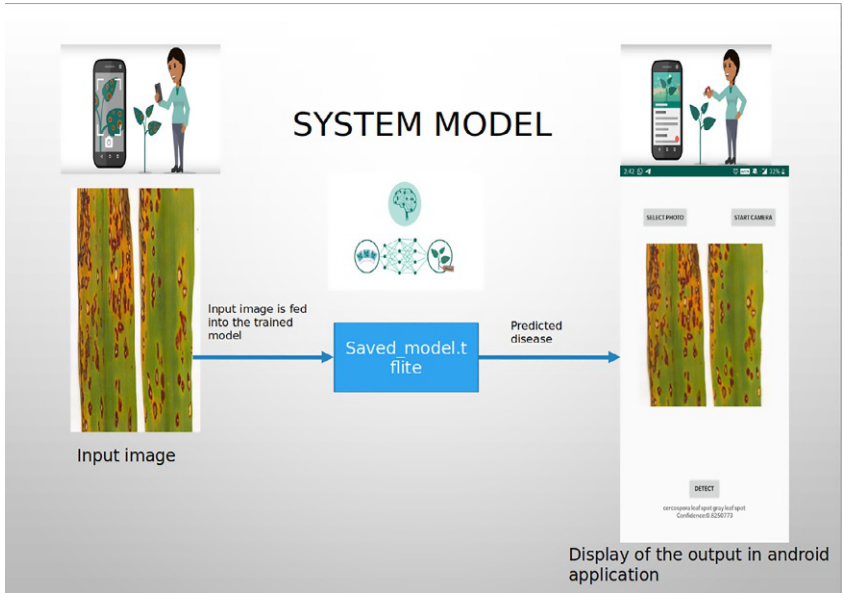


FIG. 4 SAFAL-FASAL system model.

The proposed LDD model is deployed in the Android platform for classifying the image on the phone to provide it as an end system for users. For deployment to the Android system, the tensorflow or keras model is converted to TFLite format, which is fit to be used in Android or iOS platform. Inference is performed using TFLite Android Support Library and TFLite Java API. TFLite Task Library contains a set of powerful and easy-to-use task-specific libraries for the app developers to create a ML experience with TFLite. It supports the common data formats for inputs and outputs, including images and arrays. It also provides pre- and post-processing units that perform tasks such as image cropping and resizing.

TFLite is comprised of a runtime on which any preexisting models can be run and a suite of tools used to prepare the models for use on mobile and embedded devices. Fig. 5 shows the architecture of TFLite API. A trained TensorFlow model is fed into the TFLite converter for conversion from "model.protobuf" to "model.tflite" format. This model.tflite can be deployed easily for use on Android or iOS phones.

After training the model, there are three files generated: Graph Def (.pb), Checkpoints (.ckpt), and the Saved model (.pb). The checkpoint and the graph file are frozen and optimized for inference before conversion. And finally, they are passed through the TFLite converter for conversion. All these processes are clearly represented in the Fig. 6.

2.5 Method of evaluation

Performance of the developed LDD model is evaluated using confusion matrix, precision, accuracy, recall, specificity, and F1-score as an evaluation criterion.

The confusion matrix is the way of visualizing the performance of the prediction model. Each entry in a confusion matrix denotes the number of

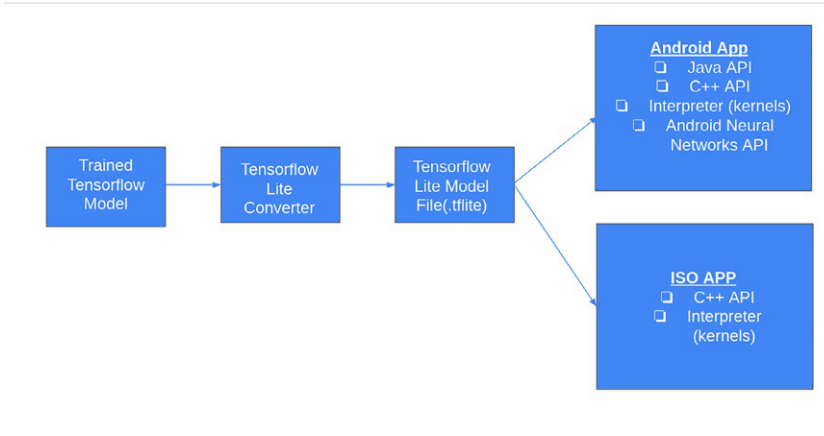


FIG. 5 The architecture of TFLite API.

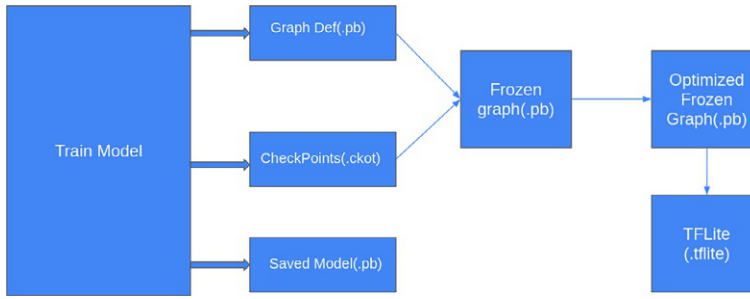


FIG. 6 TFLite converter architecture.

predictions made by the model whether it classified the classes correctly or incorrectly. Confusion matrix uses the values True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) to evaluate the correctly and incorrectly classified classes. A pixel is considered to be TP if its predicted and actual class are positive. TN is when the predicted and actual value are negative. A pixel is considered to be FP if it is wrongly predicted as positive and FN when the predicted value is wrongly predicted as negative. TP, TN, FP, and FN values are used to calculate the performance measures of precision, recall, accuracy, specificity, and F1 score, which enables us to compare the developed model against comparison models and which are computed as follows:

$$\text{Precision} = \text{TP}/(\text{TP} + \text{FP})$$

Recall, commonly called sensitivity, corresponds to the TP rate of the considered class and is computed as:

$$\text{Recall} = \text{TP}/(\text{TP} + \text{FN})$$

Specificity, it corresponds to the TN rate of the considered class:

$$\text{Specificity} = \text{TN}/(\text{TN} + \text{FP})$$

F1-score combines the precision and the recall into a single measure:

$$\text{F1} - \text{score} = 2 \times \text{Precision} \times \text{Recall}$$

$$\text{Precision} + \text{Recall}$$

The accuracy is calculated as the sum of correct classifications divided by the total number of classifications.

3 Experimentation

The experiment is performed using Python as its scripting language because of its learning simplicity and its huge collection of libraries. The dataset consists of 2490 images; these images were categorized into 6 folders with each folder

named with the class name and consisting of images belonging to the respective class. The training image consists of only RGB channels. The images in the dataset are of different resolutions, and it is necessary to convert them into a common resolution before giving it to the network for training. All the images were reduced to 255x255x3 size. The images are then converted into Numpy arrays and given to the network. The network consists of 9 layers, the combination of three convolution, activation, and max pooling layers. The experiment was performed with sparse categorical cross-entropy loss function. We run the network for 15 epochs. Initially, the loss will be high and the accuracy will be less, as the epochs increase, the network learns(extracts) more features and the accuracy increases. At the end of the fifteenth epoch, we achieved (attained) a maximum accuracy of 96% and least loss of 0.289. The corresponding graph representing the variation of loss and accuracy with the number of epochs is shown in Fig. 7. For optimization, we use Adam. The models are implemented using tensorflow-keras API. The model was trained on a system with Nvidia 2080 ti GPU.

A single image is considered to explore the feature maps across the layers, and the same is discussed here. The LDD model uses nine layers to classify the leaf diseases.

The first layer is the convolutional layer and 32 filters being applied on the $255 \times 255 \times 3$ dataset with the kernel size of 5×5. Each filter is independently convolved with the original image, and we end up with 32 feature maps (activation maps) of shape $255 \times 255 \times 3$. Fig. 8 shows the feature extraction of the input sugarcane leaf image.

The second layer is the activation function, which removes every negative value from the filtered images and replaces them with zeros. It is happening to avoid the values from adding up to zero. Rectified Linear unit (ReLU) is used as a activation function, as it only activates a node if the input is above a certain

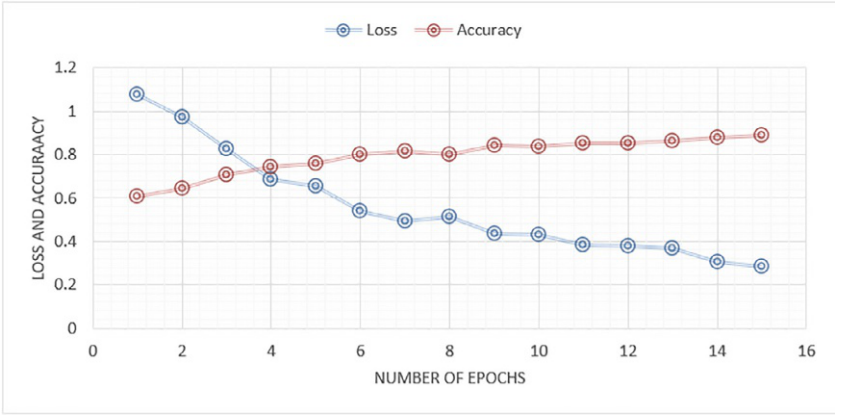


FIG. 7 Loss vs. Accuracy graph.

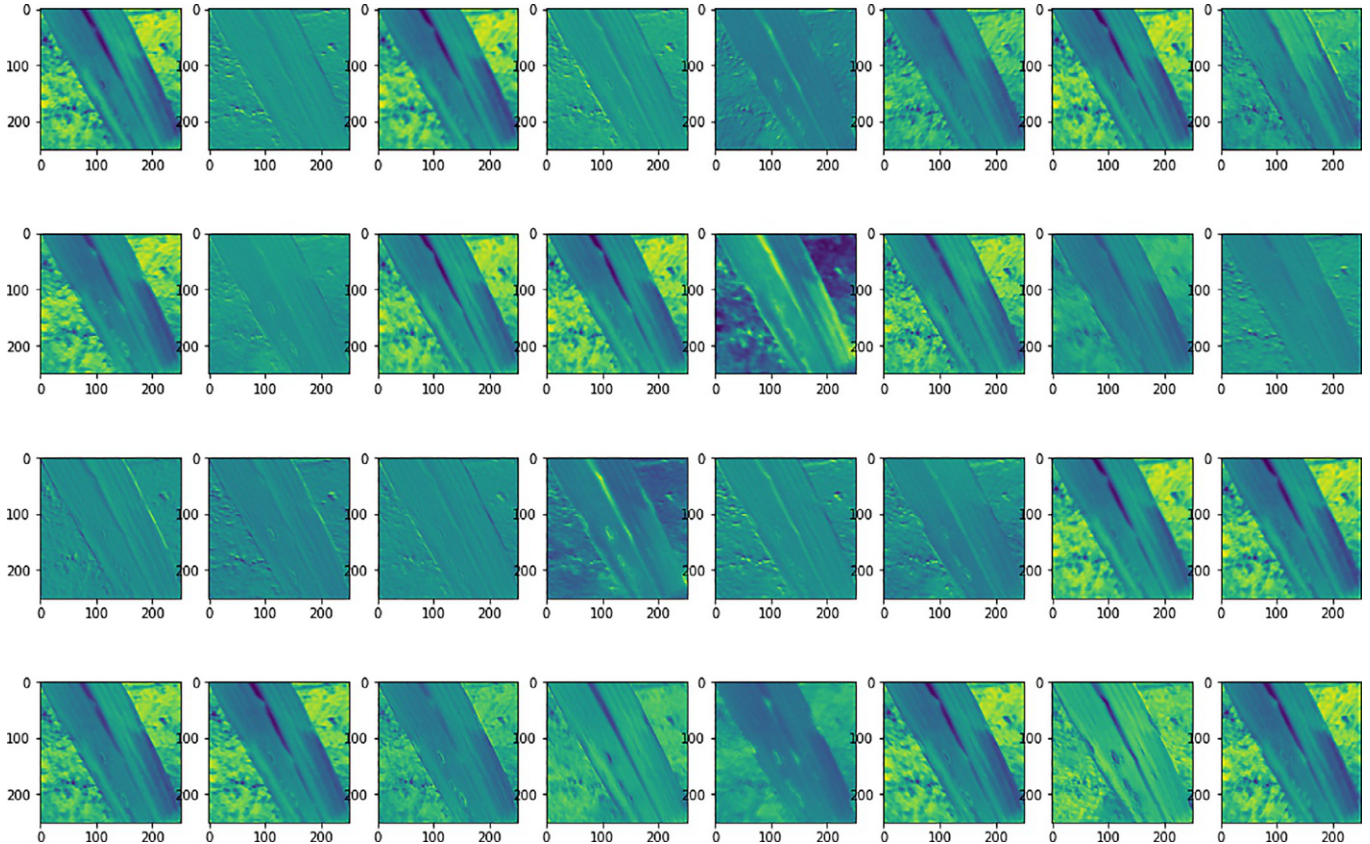


FIG. 8 Layer 1 feature maps.

threshold value. While the data is below zero, the output is zero, but the information rises above a threshold when it is above zero. It has a linear relationship with the dependent variable. The feature maps for layer 2 is given in Fig. 9.

Layer 3 is a max pooling used as a pooling layer. The input for layer 3 is the output of the previous layer (layer 2), feature maps of size $255 \times 255 \times 3$ in a deep CNN and 2×2 filter. Pooling layer operates on each feature map independently. It progressively reduces the $255 \times 255 \times 32$ feature maps to $127 \times 127 \times 32$ feature maps to reduce the number of parameters and computation in the network, as shown in Fig. 10.

In layer 4, the same operations are performed as in layer 1. Here, 64 filters are applied on the feature maps $127 \times 127 \times 32$, which is the output of layer 3 (pooling layer). This again extracts many features left out in the previous process, and we end up with 64 feature maps of shape $2127 \times 127 \times 32$, as shown in Fig. 11.

The resultant convolution feature map is passed through the ReLU activation function in layer 5 to remove the negative values from it. The resultant image of sugarcane leaf after ReLU layer 5 is shown in Fig. 12.

In layer 6, the same operations are performed as in the layer 3. Here the feature maps from the previous layer (ReLU_layer-6) are reduced from $127 \times 127 \times 64$ to $63 \times 63 \times 64$ as shown in Fig. 13.

In layer 7, the reduced image is again passed for the third iteration into the convolution layer with 64 filters and 3×3 kernel size. Now the final convolution feature map generated in the process is shown in Fig. 14.

Layer 8 passes the feature map through ReLU activation function, which takes care of removing the negative values from the feature matrix, and the resultant image is as shown in Fig. 15.

Layer 9 is the max pooling layer, which reduces the feature maps from $63 \times 63 \times 64$ to $32 \times 32 \times 64$, as shown in Fig. 16, suitable for passing through the fully connected layer for output mapping and for producing the predictions.

4 Results and discussion

This section presents and discusses the results obtained from the proposed LDD system. To verify the performance of the proposed system, the network is tested with the unseen leaves. Six images for six considered diseases were given to the network to predict their class. Fig. 17 shows the results obtained from the model along with their actual class.

Fig. 17 shows the results obtained from the trained model for the given input image. The input image of size 255×255 is given to the model as input, and the trained model processes the image and predicts the disease of the leaf in the input image. Fig. 17A depicts that the input image to the model is from the class

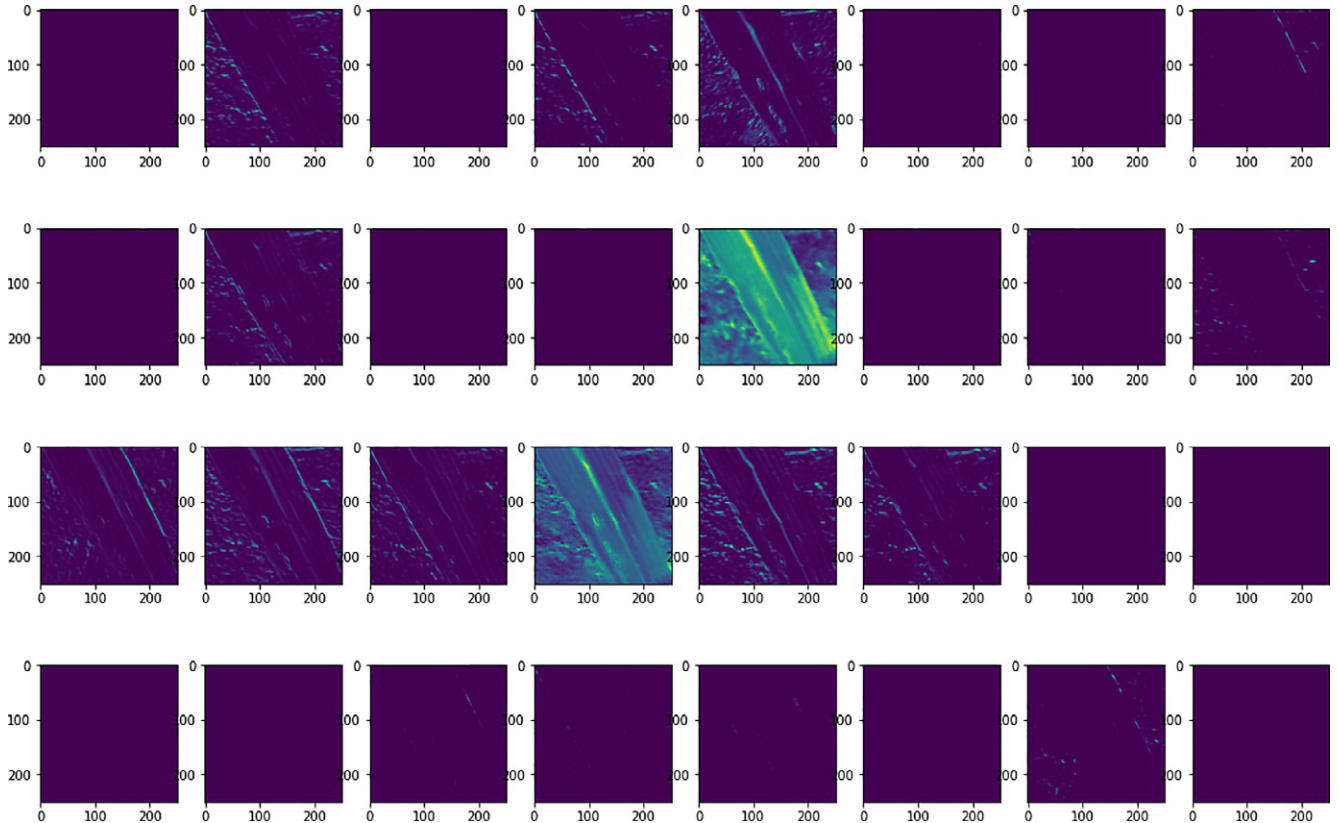


FIG. 9 Layer 2 feature maps.

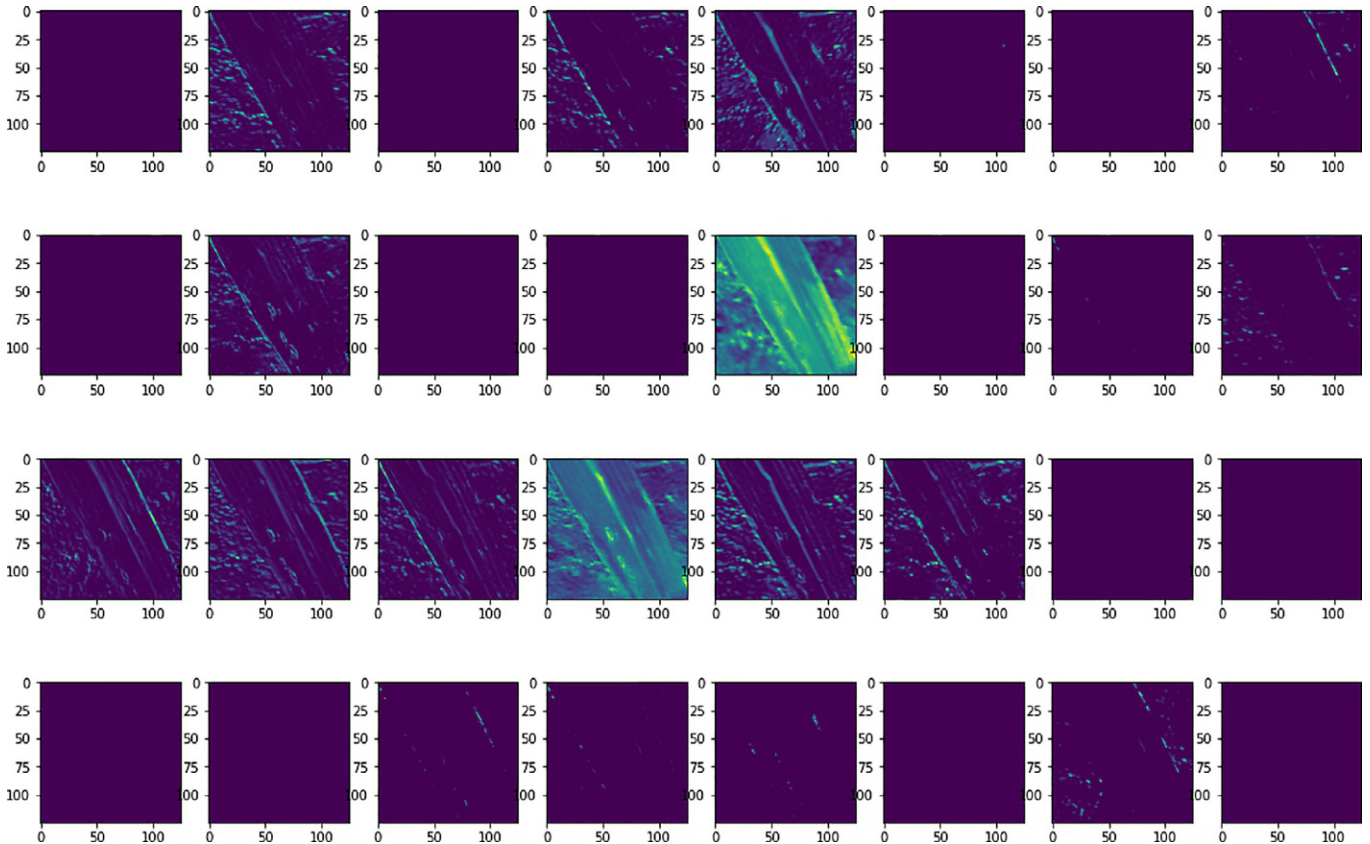


FIG. 10 Layer 3 feature maps.

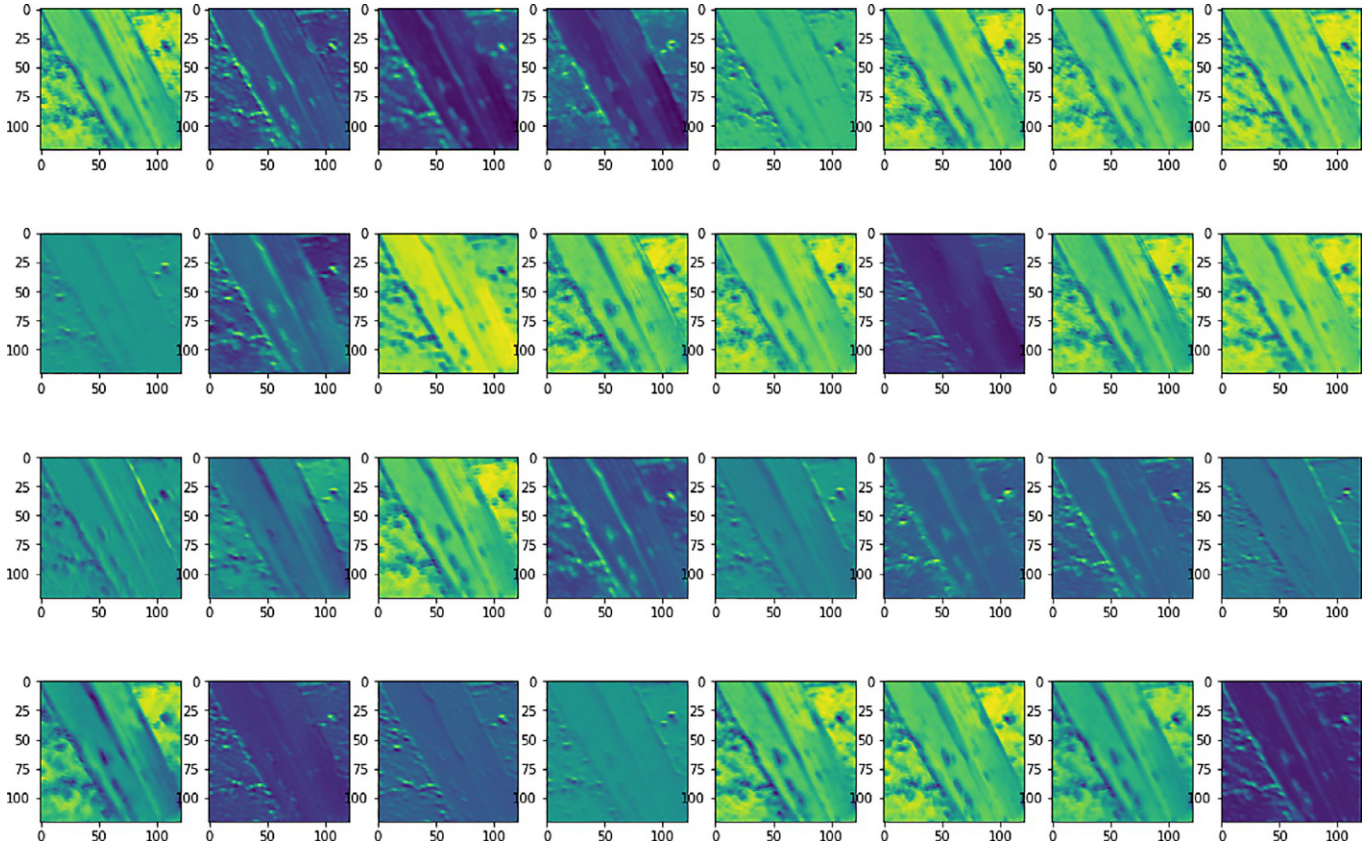


FIG. 11 Layer 4 feature maps.

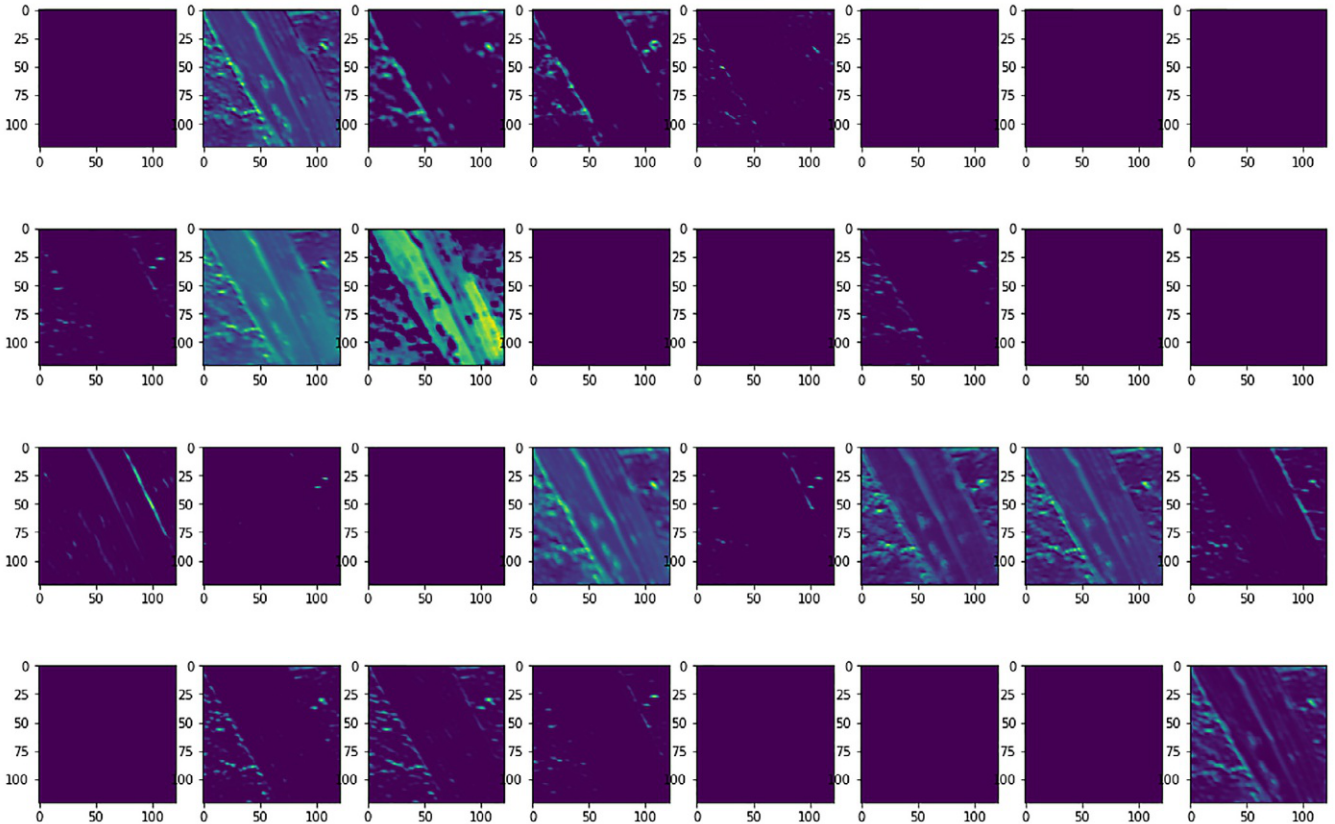


FIG. 12 Layer 5 feature maps.

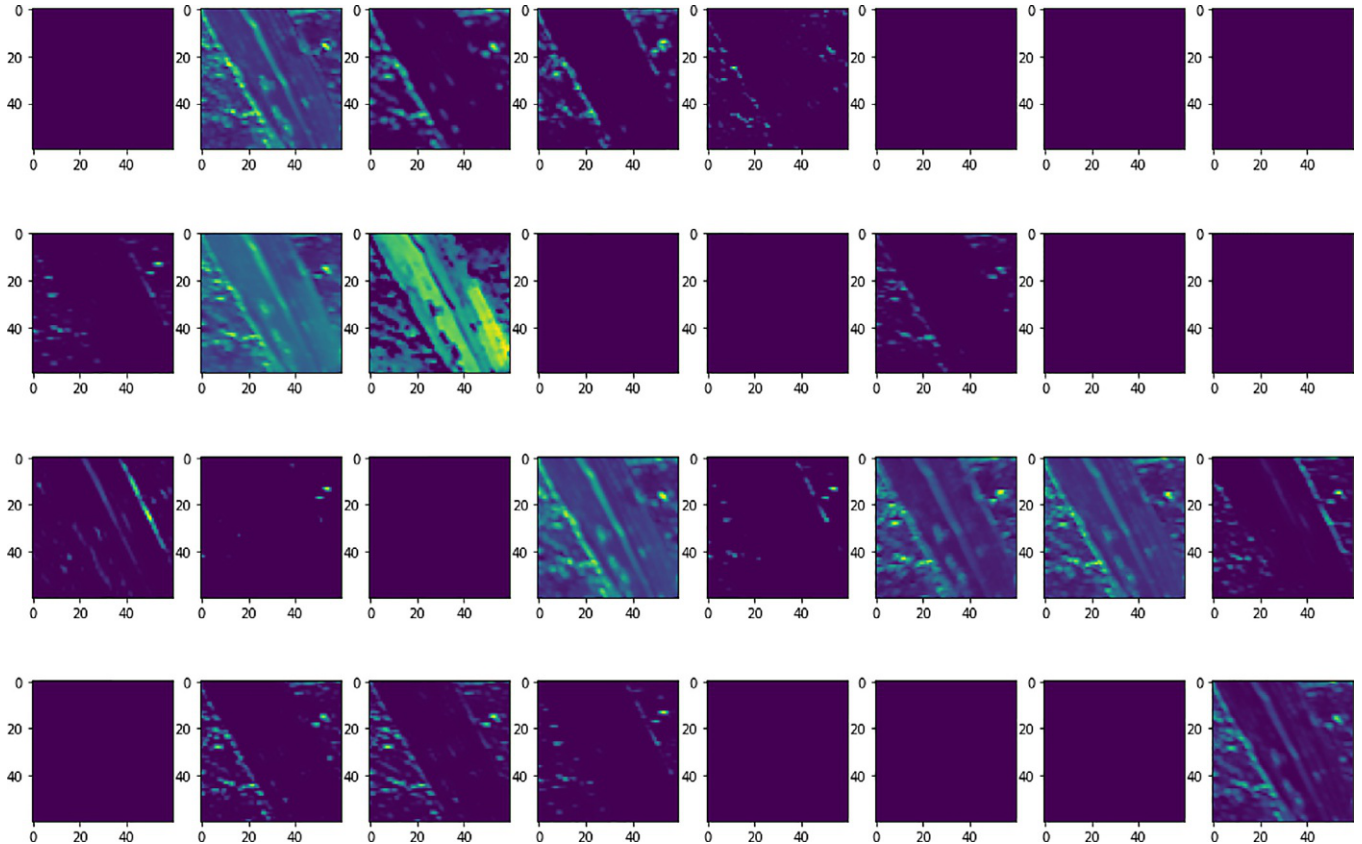


FIG. 13 Layer 6 feature maps.

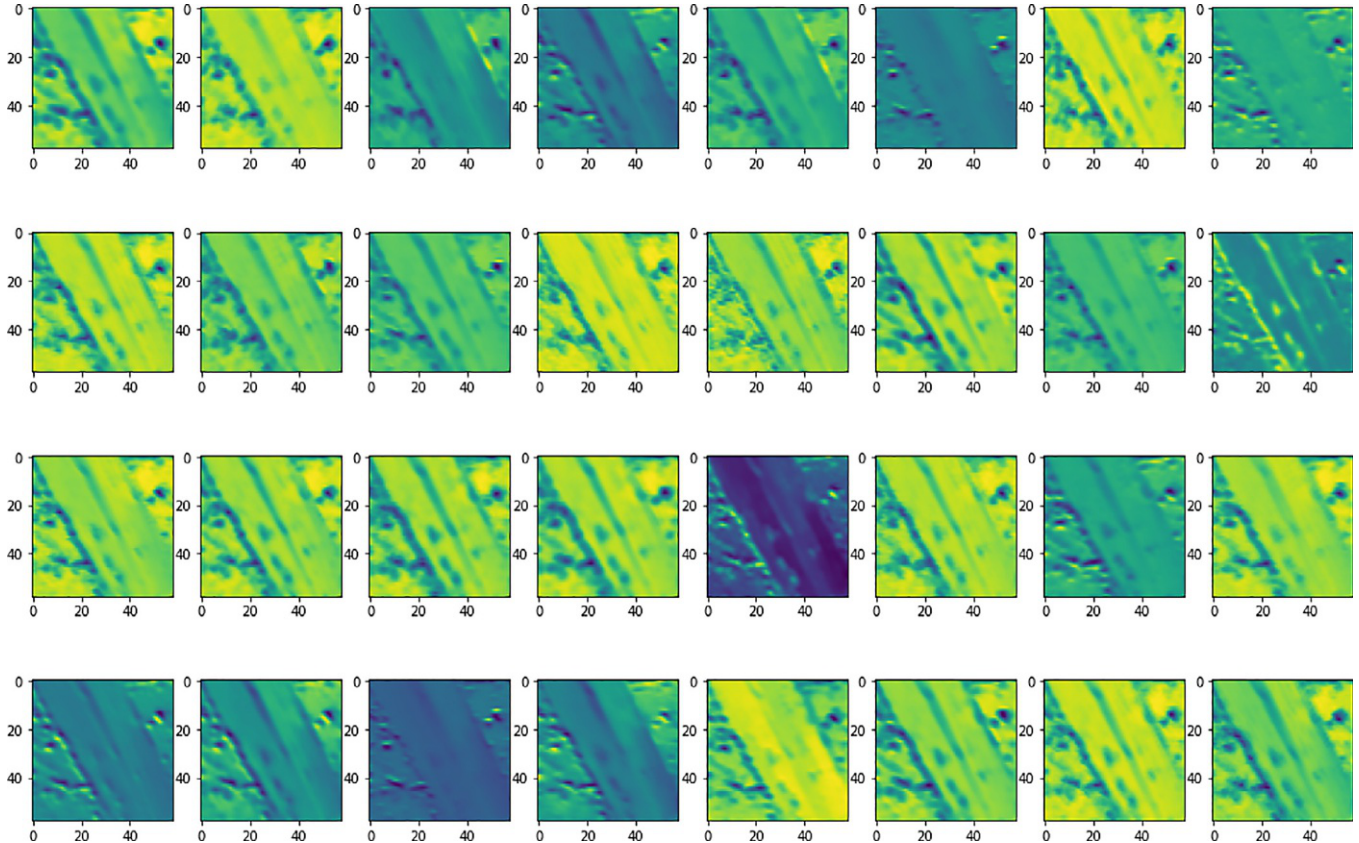


FIG. 14 Layer 7 feature maps.

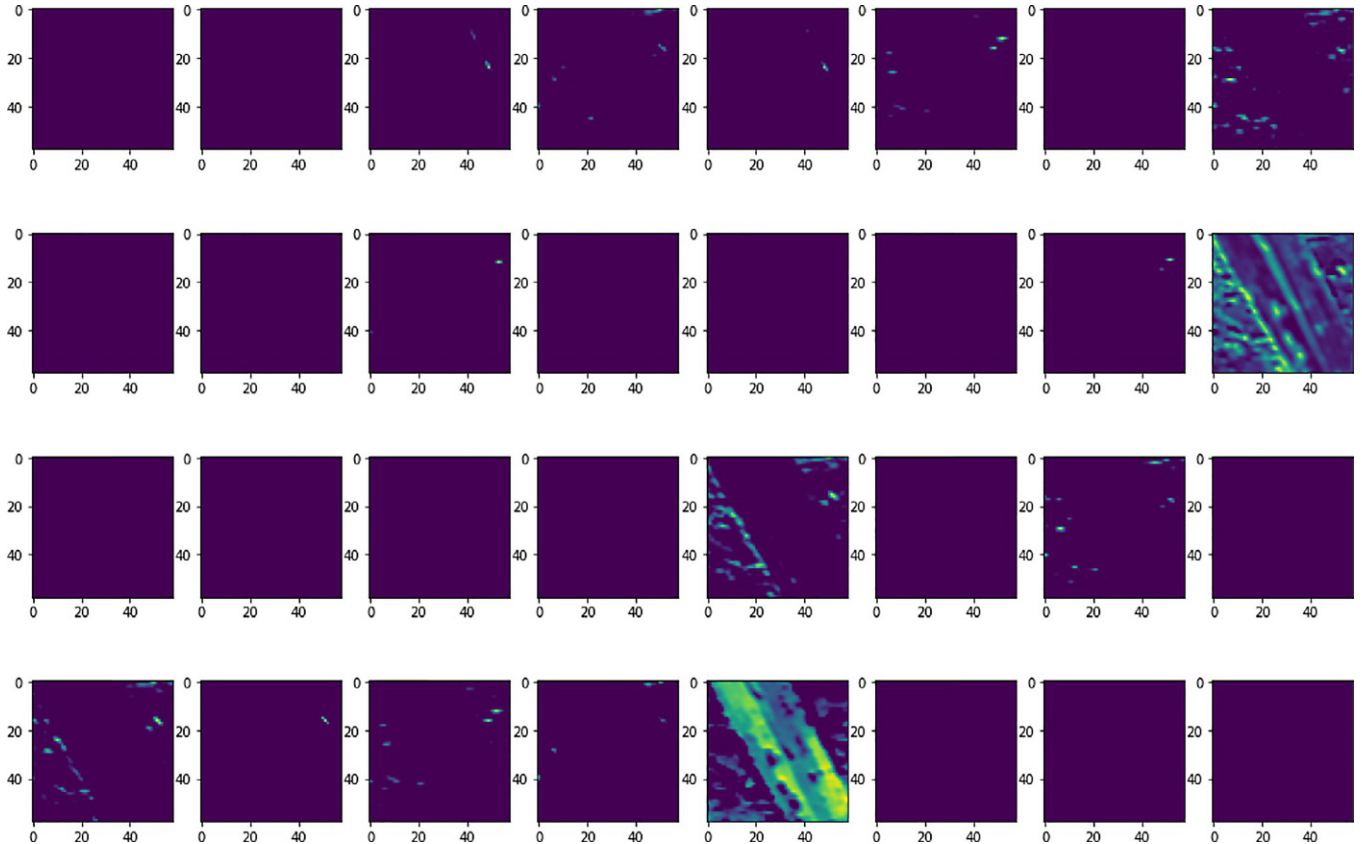


FIG. 15 Layer 8 feature maps.

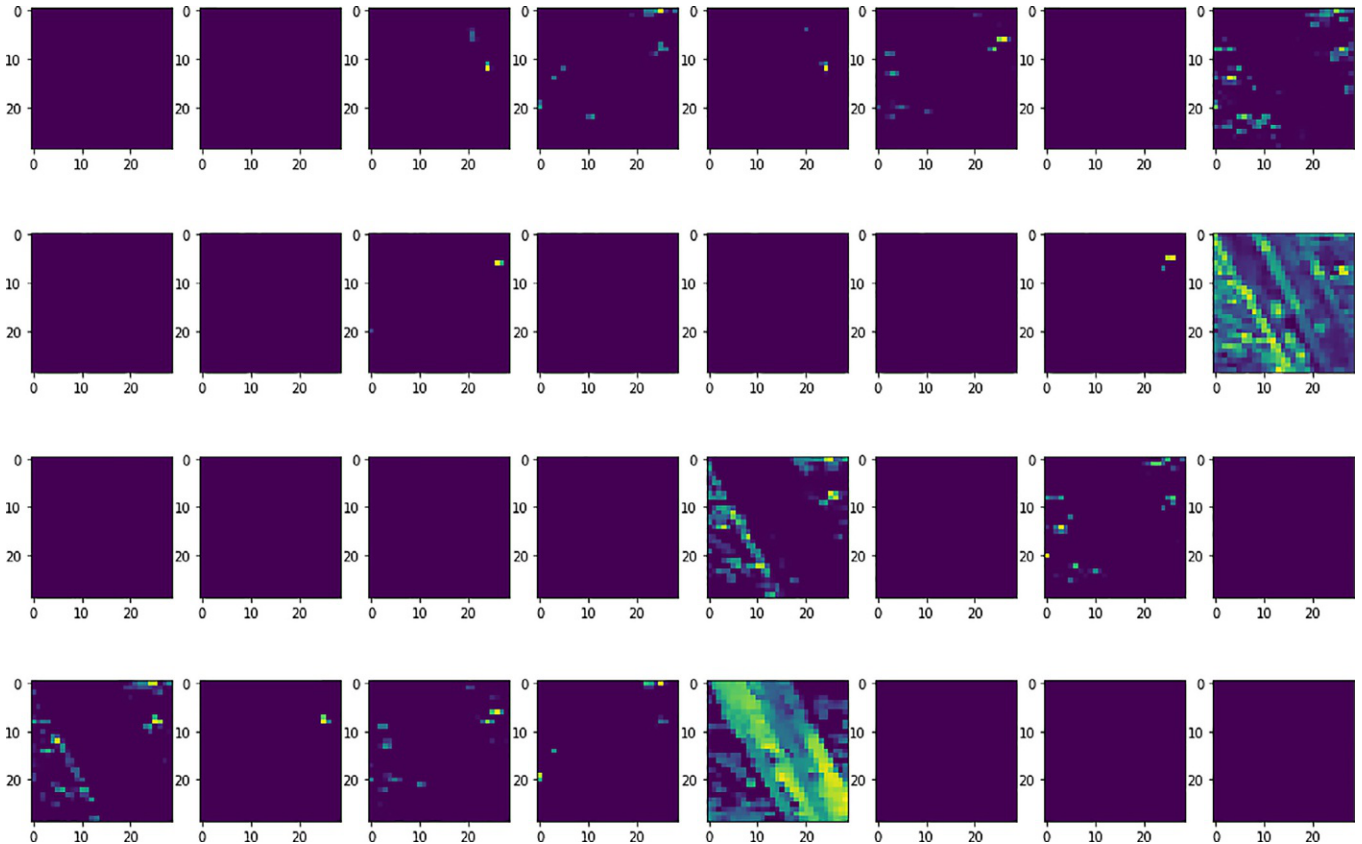
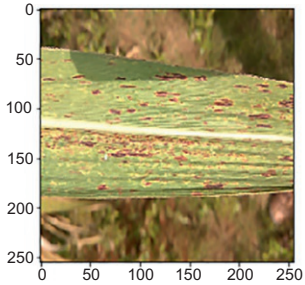


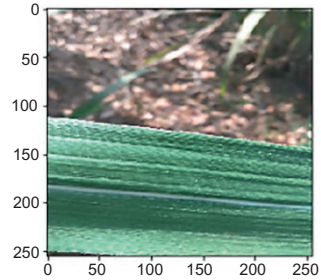
FIG. 16 Layer 9 feature maps.



Actual:cercospora-leafspot

predicted: cercospora-leafspot

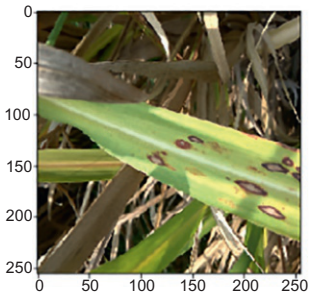
(A)



Actual:Healthy

predicted: healthy

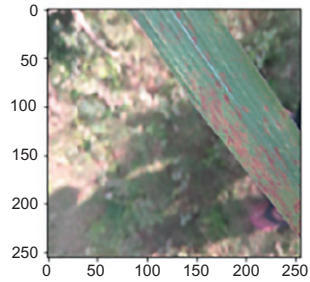
(B)



Actual:helmanthospura-leafspot

predicted: helmanthospura-leafspot

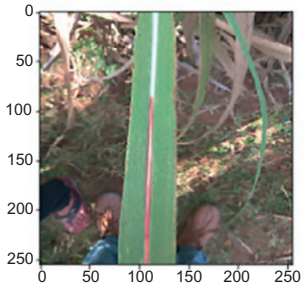
(C)



Actual:rust

predicted: rust

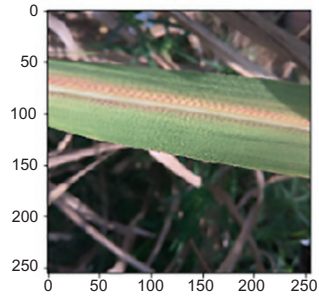
(D)



Actual:redrot

predicted: redrot

(E)



Actual:yellow-leaf-disease

predicted: yellow-leaf-disease

(F)

FIG. 17 Classification results obtained from the LDD model: (A) Cercospora, (B) Helmanthospura, (C) rust, (D) red rot, (E) yellow leaf disease, and (F) healthy.

Cercospora leaf spot, and the class predicted by the model is Cercospora leaf spot. Fig. 17B–F show the predictions of class healthy, Helmanthospura leaf spot, rust, red rot, and yellow leaf disease, respectively. It is evident that the model is performing well in predicting the classes for test images. The model is able to predict the classes accurately.

4.1 Performance evaluation

The quantitative analysis of the LDD model is evaluated using quantitative measure. We have a dataset that has six class labels: Cercospora, Helmanthospura, Rust, Red rot, Yellow leaf disease, and Healthy. Table 2 is a possible confusion matrix for these classes.

Consider the class Cercospora, the values of the metrics from the confusion matrix is calculated as follows:

$$\begin{aligned}\mathrm{TP} &= 66 \mathrm{FN} = (2+1) = 3 \mathrm{FP} = (2+2+1) = 5 \mathrm{TN} \\ &= (79+1+73+1+86+1+1+2+2+81+1+2+2+3+177) = 511\end{aligned}$$

Similarly, we can calculate the performance measures for the other classes. Table 3 is a table that shows the values of each performance measure for each class.

Since we have all necessary metrics for class Cercospora from the confusion matrix, we can now calculate the performance measures for class Cercospora as:

$$\mathrm{Precision} = 66/66+5 = 0.92$$

$$\mathrm{Recall} = 66/66+3 = 0.95$$

$$\mathrm{Specificity} = 511/511+5 = 0.99$$

$$\mathrm{F1-score} = 2 \times \frac{0.92 \times 0.95}{0.92+0.95} = 0.93$$

4.2 SAFAL-FASAL Android application results

The working procedure along with the resultant snapshots for the usage of the SAFAL-FASAL Android application is given in this section. After installing the SAFAL-FASAL application on an Android phone, a user has to open the application and select the Start Camera option to capture the leaf image through the phone camera or select the Select Photo option to select the leaf image from the phone gallery.

The captured image or the selected image will be displayed below the options, as shown in Fig. 18. The user has to select the Detect option, as shown in Fig. 19 to find the disease, which is affecting the sugarcane plant. The image will be given to the trained model for processing, and the model will give the predicted disease class name, which will be displayed below the Detect option along with the confidence. Figs. 20 and 21 are the screenshots of the application showing predictions for two classes.

TABLE 2 Confusion matrix.

Class	Cercospora	Helmanthospura	Rust	Red rot	Yellow leaf disease	Healthy
Cercospora	66	2	1	0	0	0
Helmanthospura	2	79	1	0	0	0
Rust	2	0	73	0	1	0
Red rot	0	1	0	86	1	2
Yellow leaf disease	0	0	2	0	81	1
Healthy	1	0	2	2	3	177

TABLE 3 Precision, recall, specificity, F1 score matrix.

Class	Precision	Recall	Specificity	F1-score
Cercospora	0.92	0.95	0.99	0.93
Helmanthospura	0.96	0.96	0.99	0.96
Rust	0.92	0.96	0.98	0.93
Red rot	0.97	0.95	0.99	0.95
Yellow leaf disease	0.94	0.96	0.99	0.94
Healthy	0.98	0.95	0.99	0.96

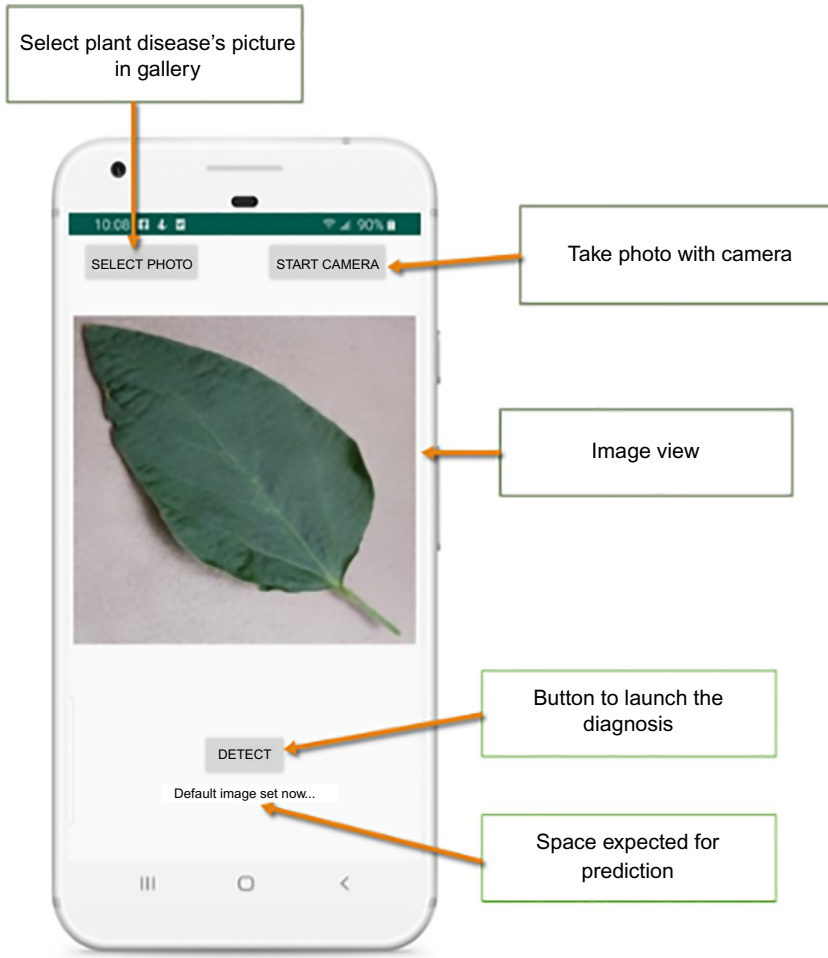


FIG. 18 Selecting an image.

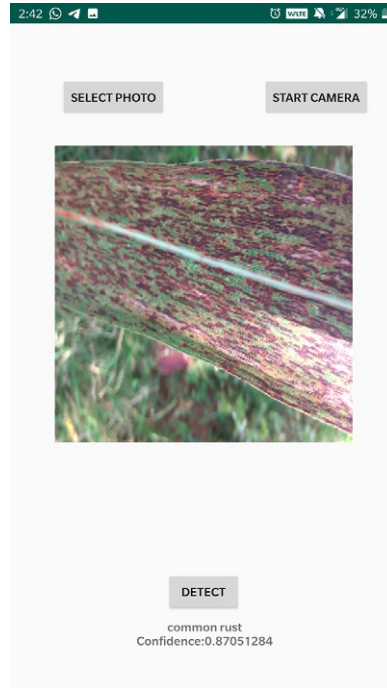


FIG. 19 Uploading the image for disease detection.

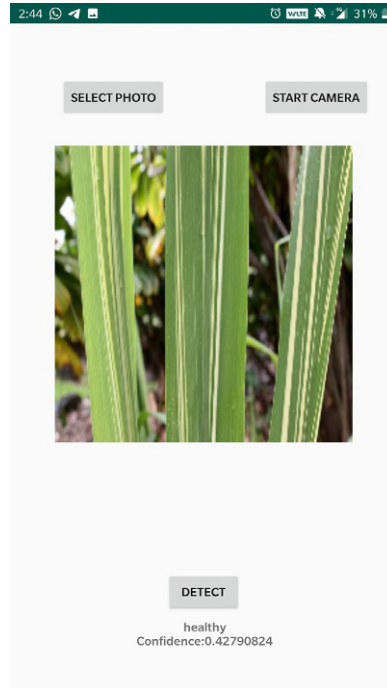


FIG. 20 Predicted output 1.

FIG. 21 Predicted Output 2.

5 Conclusion

Many farmers spend money on disease management without adequate technical support. This results in poor disease management and control. The proposed approach helps farmers detect disease more correctly with less effort. It reduces the time required to detect diseases. The proposed LDD model achieves 98% of accuracy in detecting the disease classes. So, this can be used as a cost-effective solution to our farmers to address the issues of sugarcane leaf detection.

References

Kulkarni, O. (2018). Crop disease detection using deep learning. In *Proceedings of 2018 4th International conference on computing communication control and automation ICCUBEA 2018* (pp. 2018–2021). https://doi.org/10.1109/ICCUBEA.2018.8697390.

Maniyath, S. R., et al. (2018). Plant disease detection using machine learning. In *Proceedings of 2018 international conference design innovations for 3Cs compute communicate control ICDI3C 2018* (pp. 41–45). https://doi.org/10.1109/ICDI3C.2018.00017.

Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*, *7*, 1–10. https://doi.org/10.3389/fpls.2016.01419.

Padilla, D. A., Magwili, G. V., Marohom, A. L. A., Co, C. M. G., Gaño, J. C. C., & Tuazon, J. M. U. (2019). Portable yellow spot disease identifier on sugarcane leaf via image processing using support vector machine. In *2019 5th international conference on control, automation, and robotics ICCAR 2019* (pp. 901–905). https://doi.org/10.1109/ICCAR.2019.8813495.

Patil, M. A. N., & Pawar, M. V. (2017). Detection and classification of plant leaf disease. *Iarjset*, *4* (4), 72–75. https://doi.org/10.17148/iarjset/nciarcse.2017.20.

Saleem, M. H., Potgieter, J., & Arif, K. M. (2019). Plant disease detection and classification by deep learning. *Plants*, *8*(11). https://doi.org/10.3390/plants8110468.

Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., & Stefanovic, D. (2016). Deep neural networks based recognition of plant diseases by leaf image classification. *Computational Intelligence and Neuroscience*, *2016*. https://doi.org/10.1155/2016/3289801.

Toda, Y., & Okura, F. (2019). How convolutional neural networks diagnose plant disease. *Plant Phenomics*, *2019*, 1–14. https://doi.org/10.34133/2019/9237136.

This page intentionally left blank