International Journal of Computational and Electronic Aspects in Engineering
Volume 3, Issue 2, June 2022, pp 32 - 38
https://doi.org/10.26706/ijceae.3.2.arset1002

Plant Disease Prediction Using Deep Learning

J. Jyotsna[1]
jyotsnaj136@gmail.com

Prachi Ramteke[2]
prachipratiksha04@gmail.com

Prity Baxla[3]
bitrkeytnd@gmail.com

Department of Computer Science Engineering, Christian College of Engineering and Technology, Bhilai, India

***Abstract* — This study aims to develop an android application to detect and identify plant diseases through deep convolutional neural network. Plant disease is a critical issue in agricultural countries like Bangladesh. Every year production of crops sustains heavy loss due to diseases. It is quite difficult to detect plant diseases with human eyes. So it is essential to build an automated system to detect the diseases. The proposed disease detection model takes an image of a plant leaf as input, processes it and uses deep convolutional neural network to detect and identify the disease. The developed mobile application is user-friendly and can be used by farmers without much technical knowledge.**

Keywords: Deep Convolutional Neural Network, Image Processing, Android Application, Plant Disease Detection.

I. INTRODUCTION

Plant diseases are one of the grand challenges that are faced in the agriculture sector worldwide. The farmers and other plantation growers do not possess the expertise and resources to correctly identify the diseases of plants and their remedies. To handle this problem machine learning technology can be used, which can correctly identify the disease of the plants and display the remedies to the end user.

According to past studies, 42% of agricultural production is in loss and that too only because of the increasing rate of loss due to plant leaf diseases. Recent advancements in technology have created remarkable opportunities in developing many countries. Android mobile phones are now cheap and affordable for lower-earning people. An automated system can greatly help the farmers to diagnose crop diseases easily and take actions accordingly to avoid the waste of crops. Hence, we got motivated to create an android application that can detect the diseases of plants from captured images of leaves.

Researchers around the world have taken the plant disease problem seriously and are searching for ways to prevent plant diseases and detect them at an early stage. Many researchers have approached various techniques to identify plant diseases. Image Processing, Machine Learning, Deep Learning are some of the techniques being used to try to solve the problem. By using Tensor-Flow, the models trained using deep convolutional neural network (CNN) can be used in wide varieties of mobile devices, even low-end devices.

Tensor flow lite models use very little hardware resources which is why it is supported in most mobile devices. This study aims to develop a system for the detection of plant diseases through deep CNN and image processing. Furthermore, the study aims to use the developed model to create a user-friendly android application to detect plant disease from images captured with the camera of the phone and give solutions to cure the disease. Plants show a range of symptoms such as colored spots, or streaks occurring on the leaves when they are diseased. As the disease progresses, the visual symptoms change their color, shape, and size. With the help of deep

Technical Article
Available online on – 08 August 2022

Cite this article – J. Jyotsna, Prachi Ramteke, Prity Baxla, "Plant Disease Prediction Using Deep Learning", *International Journal of Computational and Electronic Aspects in Engineering*, RAME Publishers, vol. 3, issue 2, pp. 32-38, 2022.
https://doi.org/10.26706/ijceae.3.2.arset1002

CNN, we can train these patterns and create a model to recognize them.

By using the application, farmers can capture the images of plant leaves using any mobile camera of average quality. The image is then processed and cross-matched with the integrated model to identify the disease and provide solutions based on the detected disease. Thus, the farmers can save their time and money as well as their crop

II. Related Literature and Studies

The literature survey is conducted to compare different methodologies previously proposed for identifying plant diseases using deep learning and image processing. Many studies have proposed different solutions to detect diseases. In [4], an IoT-based system was presented that can automatically detect and identify plant leaf diseases. The authors used sensor devices to collect images of plants and plant leaves. Image processing, k-means clustering algorithm, and artificial neural networks are used. The invented device classifies diseases based on monitoring the temperature, humidity, and moisture of the plant with an accuracy over 90%.

In [5], the authors proposed a system that uses an edge detection technique to detect the diseased zones of the plant or fruit. Images of the fruit are captured first. Then, image segmentation is done using a segmentation technique. Afterward, the edges of the diseased zones are calculated in pixels. Depending on the number of pixels, the rate of contamination in the plant or fruit is provided. The control and treatment methods are provided hinged on the affected disease of the fruit.

In [6], deep learning neural network algorithm was explored. Tensorflow is used to process the data to be usable for training. Through the use of deep learning and neural network algorithm and based on the F1 score, the model for detecting plant diseases is built. An application was implemented and evaluated by specialists in this area of study. The accuracy of their developed model is mentioned to be 80%.

In [7], different kinds of deep learning model architectures were implemented, which are based on CNN architectures, to identify plant diseases with leaf images of healthy or diseased plants. VGG CNN architecture performed the best. It accomplished the accuracy of approximately 99% in classifying 17,548 plant leaf images.

In [8], the authors focused on recognizing a paddy plant disease using image processing. The system takes the paddy leaf image as an input and converts the RGB image to grayscale. Then the morphological opening operation is applied to reduce noise and finally image segmentation. After these processes, they find the infected region of the paddy leaf.

In [9], a mobile application was developed for plant disease recognition using image processing which analyzes the color patterns of the diseased marks in plant leaves and bodies. The dataset images were captured under laboratory conditions with the help of a digital camera. The distance matrix is employed to calculate the distance between each pair of species. Image segmentation is used to partition the image of a plant into distinct regions containing each pixel with similar attributes. Mainly, the k-means clustering algorithm is implemented to identify the diseases. The authors claimed to have achieved 90% accuracy for their model using a small training set.

In [10], the authors showed effective and correct plant disease detection and identification techniques through the use of image processing in MATLAB. K-means clustering algorithm and multi–Support Vector Machine (SVM) methods are used which are organized for both plant and fruit disease identification. Image segmentation and feature extraction are used to prepare the images for training.

In [11], the authors presented a study on various disease identification methods that are utilized in detecting plant diseases. They also described a method for image segmentation which is used to detect and identify plant diseases. The accuracy of the presented system is described to be approximately 95%. But, the number of sample images used in the system is only 60 for 4 different species.

In [12], a model is trained on images of plant leaves using the deep CNN with the goal to classify both crop species and the identity of the diseases. The proposed model

can classify very quickly which is ideal for implementing into an application. However, the accuracy gained on the images which are not from the dataset is mentioned to be just above 31%.

In [13], different procedures to segment the diseased area of the plants were explained. The authors studied various feature extraction and identification methods that are used for extracting the features of the diseased leaf, additionally identifying the plant diseases. The utilization of artificial neural network methods for the detection of plant diseases like back-propagation algorithm and SVM are discussed.

II. METHODOLOGY

A. *System Overview*

The overview of the proposed system is shown in figure 1. The user of the application captures the image of a leaf of the subject plant using the phone camera. The captured image is then processed using the trained model integrated into the mobile application. The application then gives results based on the accuracy from the image whether the plant is healthy or diseased. The result will show the specific disease name as well as the solutions to cure the disease. There is also a feature to call the nearest agriculture department if the situation is too worse to handle by the user alone.

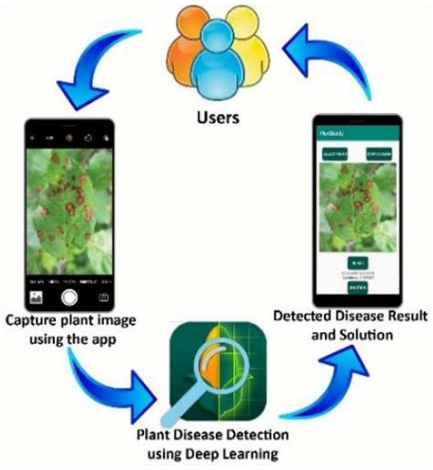

Figure 1: Overview of the proposed system

B. *System Development Method*

The development method of the system is depicted below. It shows the gradual steps of the system from training the model to implementing it in the application.

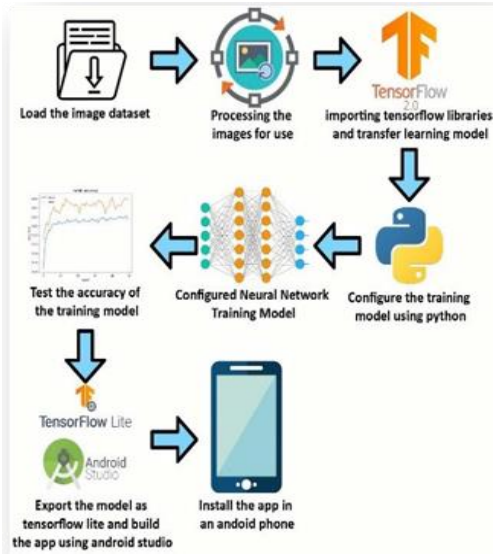

Figure 2: Development method of the system

The detailed system development steps are described as following:

a. Dataset Collection and Load

A public dataset of 54,305 images of plant leaves is utilized for training the model. The dataset is created by Plant Village and can be found in Kaggle. The resolution of the images is 256x256 pixels. The images are saved in jpeg format. The train-validation split is 80%-20%, which makes 43,456 images for training the model and 10,849 images for validation. The dataset contains images of 14 different species of crops and 26 different diseases.

Figure 3: Sample images of the datase

b. Data Augmentation

To avoid the over-fitting issue of the CNN model, we augmented the number of images for a few disease classes that lack an insufficient training set and have a lot of background noise such as tomato target spot, tomato Septoria leaf spot, and tomato late blight. We used the geometric transformations method to eliminate the positional biases present in the training data. As shown the

geometric transformations applied to these classes were horizontal flipping, −45° to 45° rotation, 1.5× scaling, filling with nearest neighbor regions, zoom with range 0.2, width and height shifts with a relative scale of 0.3, and cropping some image manually.

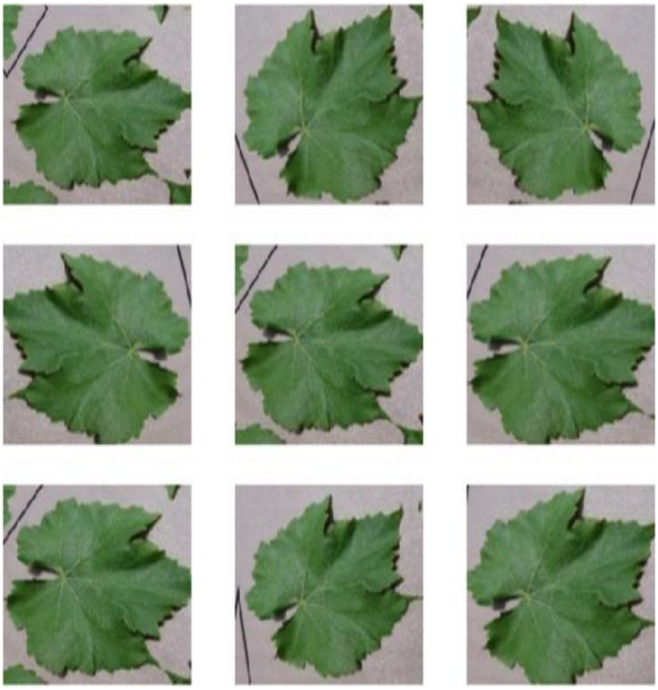

Figure 4: Geometric transformations applied to the classes

c. Image Pre-processing

Image pre-processing is employed to process digital images using algorithms to be usable for the system. It cleans the noise and distortion in images.

Data generators are set up to read images from the source folder. Data augmentation is used to process the images in Keras. This technique enhances the condition of dataset images so that deep CNN models can be built utilizing them. It applies a series of random transformations such as resizing, rotation, RGB to grayscale, mixing images, etc.

The images are then normalized in the range [0, 1] and resized to the required size for the network.

1. Implementation

This section presents the implementation details of the plant disease detector at the cloud and mobile sides.

2. CNN Implementation

The CNN model is implemented using Keras Development environment (2.4). Keras is an open-source neural network library written in Python, which uses TensorFlow-02 as a back-end engine. Keras libraries running on top of Tensor Flow make it relatively easy for developers to build and test deep learning models written in Python.

For instance, we used the keras. preprocessing. image. Image Data Generator library to augment some images in our dataset via several geometric transformations; therefore, our model would never see twice the same image. This helps to avoid over-fitting and helps the model generalize better.

The training images must have the same size before feeding them as input to the model. Our model was trained with colored (RGB) images with resized dimensions of 200 × 200 pixels.

We set the batch size and number of epochs to be 150 images and 10 epochs, respectively. The model training was carried out using a server computer equipped with a 4.50 GHz Intel Core™ i7-16MB CPU processor, 16 GB of RAM, and RTX-3060 CUDA GPU 3584-cores with a base clock speed of 1320 Mhz. The training phase took approximately 2 days to run 10 epochs. We took a snapshot of the trained weights every 2 epochs to monitor the progress. The training error and loss are calculated using this equation:

$$M = \frac{1}{n}\sum_{i=1}^{n}.(y_{i-}x_i).^2$$

where M is the mean square error of the model, y is the predicted class calculated by the model, and x is the actual class. M represents the error in object detection.

The CNN network has two types of layers: convolution and pooling. Each layer has a group of specialized neurons that perform one of these operations. The convolution operation means detecting the visual features of objects in the input image such as edges, lines, color drops, etc. The pooling process helps the CNN network to avoid learning irrelevant features of objects by focusing only on learning the essential ones. The pooling operation is applied to the output of the convolutional layers to down sampling the generated feature maps by summarizing the features into patches. Two common pooling methods are used: average-pooling and max-pooling. In this paper, we used the max-

pooling method, which calculates the maximum value for each patch of the feature map as the dominant feature.

As shown in Figure 3, the output of every Conv2D and MaxPooling2D layer is a 3D form tensor (height, width, channels). The width and height dimensions tend to shrink as we go deeper into the network. The third argument (e.g., 16, 32 or 64) controls the number of output channels for each Conv2D layer. During the training phase, the CNN model generated around 4 million trainable parameters.

Before moving the trained CNN model to the mobile device, we converted it into an optimized IR model based on the trained network topology, weights, and biases values.

Layer (type)	Output Shape	Param #
rescaling_1 (Rescaling)	(None, 180, 180, 3)	0
conv2d (Conv2D)	(None, 180, 180, 16)	448
max_pooling2d (MaxPooling2D)	(None, 90, 90, 16)	0
conv2d_1 (Conv2D)	(None, 90, 90, 32)	4640
max_pooling2d_1 (MaxPooling2	(None, 45, 45, 32)	0
conv2d_2 (Conv2D)	(None, 45, 45, 64)	18496
max_pooling2d_2 (MaxPooling2	(None, 22, 22, 64)	0
flatten (Flatten)	(None, 30976)	0
dense (Dense)	(None, 128)	3965056
dense_1 (Dense)	(None, 10)	1290

Total params: 3,989,930
Trainable params: 3,989,930
Non-trainable params: 0

Figure 5: The Structure of the CNN model.

Given our plant disease predictor model is considered a multi-class classification problem, where it classifies the input image as belonging to one or more of the 38 disease classes, we used the SoftMax activation function at the output layer cross-entropy as the loss function. Figure below illustrates the calculated training error and loss graphically. The mean squared error loss decreases over the ten training epochs, while the accuracy increases consistently. We can see that our model converged after the 8th epoch, which means that our dataset and the fine-tuned parameters were a good fit for the model.

d. Checking the Accuracy of the Model

The configured model is trained for several epochs to check accuracy. It is important to check over-fitting and under-fitting in this step, which is observed by plotting a graph of accuracy and loss of the model, shown later. To achieve the highest possible accuracy, the model is trained several times tweaking a few parameters.

e. Exporting the Model and Building the Mobile Application

In the last step, the trained model is exported to flite model and the android application, along with the features, is developed using android studio. The application is then ready to be installed in an android smartphone to use.

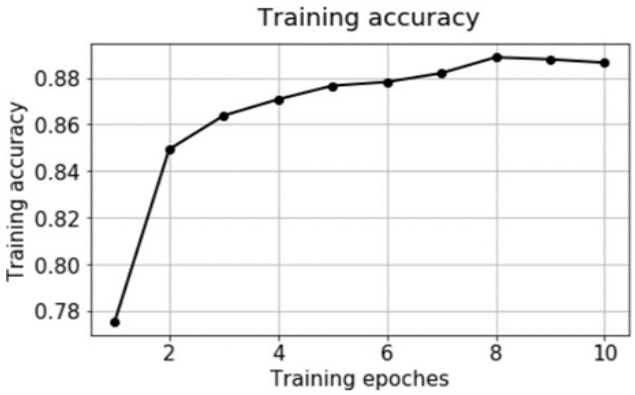

Figure 6: Training accuracy

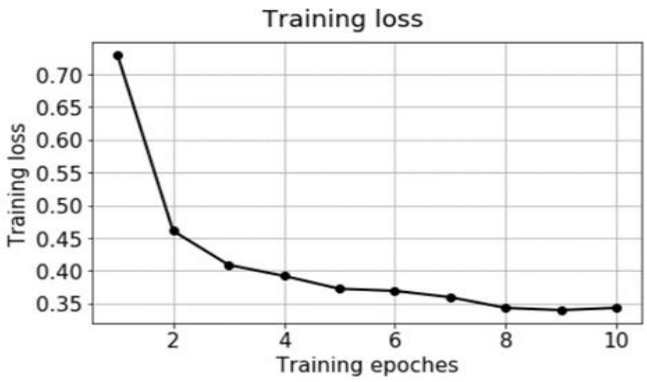

Figure 7: Training loss

f. The Calculation of Precision Recall Values of the CNN Model

The precision ratio describes the performance of our model at predicting the positive class. It is calculated by dividing the number of true positives by the sum of the true positives and false positives, as follows:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{True Positives}}$$

The recall ratio is calculated as the ratio of the number of true positives divided by the sum of the true positives and the false negatives, as follows:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

F1-score ratio is calculated by a weighted average of both precision and recall, as follows:

$$\text{Recall} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

III. Results And Performance Analysis

Training Accuracy: The percentage of successful classification in the training dataset. This means the amount of correctly identified images from the dataset.

Training Loss: The average mistakes of the model in the training dataset. This means the average loss per batch, out of batches of the training dataset.

Validation Accuracy: The percentage of successful classification of classes in the validation dataset.

Validation Loss: This indicates the average loss per batch, out of batches of the validation dataset. Model Size: It indicates the size of the exported trained model in megabytes (MB).

Accuracy of the model: The accuracy of the chosen model on 4 randomly chosen images from the validation dataset is shown below. It can be seen that in three cases, the accuracy of the detected class is above the average training accuracy which is approximately 97%. In one case, the accuracy is below the average training accuracy. This determines that the performance of the selected model is acceptable and is ready to be used to build the mobile application.

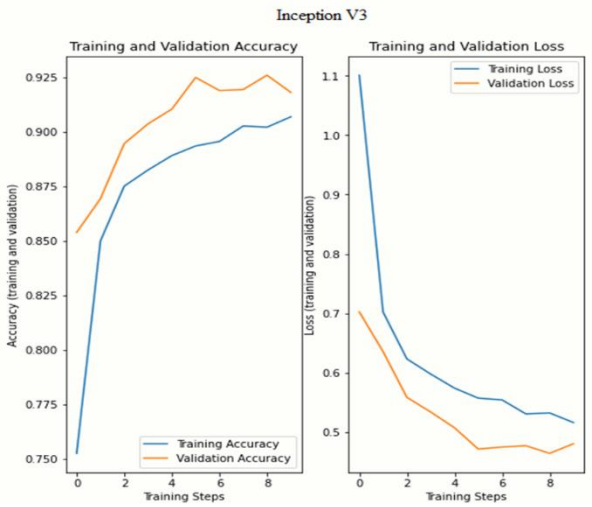

Figure 8: Inception V3 response of training vs validation accuracy and validation loss

IV. Conclusion

In this study, an android based mobile application has been developed to detect and identify plant diseases using deep CNN. The model has been trained using a large public dataset of plant images and has attained an accuracy of around 97% which indicates the acceptability of the model. The trained model takes an image of a plant leaf as input, processes it, and uses deep CNN algorithm to detect and identify plant disease. The model is exported to the android application.

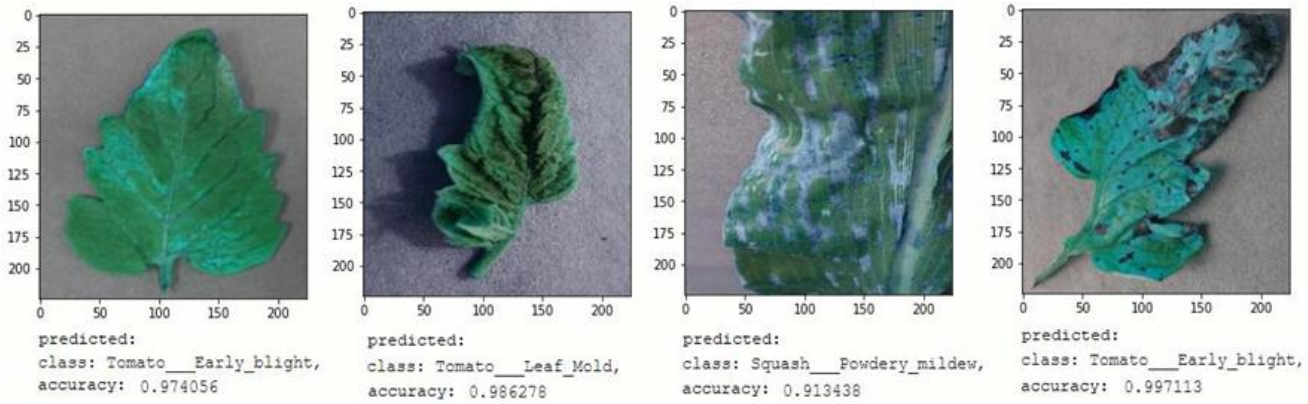

Figure 9: Checking the accuracy of the model on 4 random images.

The application also provides necessary steps to prevent and cure the diseases. Due to the availability and low price of android mobile devices in recent times, even the lower-earning people like farmers can easily afford android devices.

The proposed application of this study runs and works well in low-end devices as well. Moreover, the application interface is very easy to understand.

So, it is quite practical for the farmers to be able to use the application. Farmers in rural areas can

use it to detect plant diseases accurately and take action accordingly. It will help to avoid disaster in food production, thus increasing gross foo d production. The dataset images used in the study are captured in indoor lighting and *environment.*

Hence, the performance of the proposed model decreases around 10-20% when images are captured outdoor. This limitation can be overcome by using a dataset of images that are captured in a real environment.

This work can be further explored to add new features for making the application more user-friendly. There is also a scope to build the system u sing a different dataset or images of other species of plants. Multiple language support can make it usable in different countries.

REFERENCES

1. Kaggle: Machine Learning and Data Science Community. 2021. Available online: https://www.kaggle.com/ (accessed on 25 June 2021).
2. Tensorflow: A Machine Learning Platform. 2021. Available online: https://www.tensorflow.org/ accessed on 25 June 2021).
3. Patel, A., Joshi, M.: A Survey on the Plant Leaf Disease Detection Techniques. IJARCCE. vol. 6, pp. 229-231 (2017).
4. Thorat, N. Nikam, S.: Early Disease Detection and Monitoring LargeField of Crop by Using IoT. (IJCSIS) International Journal of Computer Science and Information Security. vol.15, pp. 236-248 (2017).
5. D M, S., Akhilesh Kumar, S., M G, R., C, P.: Image based Plant Disease Detection in Pomegranate Plant for Bacterial Blight. 2019 International Conference on Communication and Signal Processing (ICCSP). pp. 0645-0649. IEEE, Chennai, India (2019). https://doi.org/10.1109/ic-csp.2019.8698007
6. Valdoria, J.,Caballeo, A., Fernandez, B., Condino, J.:iDahon: An Android Based Terrestrial Plant Disease Detection Mobile Application Through Digital Image Processing Using Deep Learning Neural Network Algorithm. 4th International Conference on Information Technology (InCIT).pp. 94-98. IEEE, Bangkok, Thailand (2019). https://doi.org/10.1109/incit.2019.8912053
7. Ferentinos, K.: Deep learning models for plant disease detection and diagnosis. Computers and Electronics in Agriculture. vol. 145, pp. 311-318 (2018). https://doi.org/10.1016/j.compag.2018.01.009
8. Mahalakshmi, J., Shanthakumari, G.: Automated Crop Inspection and Pest Control Using Image Processing. International Journal of Engineering Research and Development. vol. 13, pp. 25-35 (2017).
9. Thakre, G., More, A., Gajakosh, K., Yewale, M., Shamkuwar, D.: A Study on Real Time Plant Disease Diagnosis System. International Journal of Advance Research, Ideas and Innovations in Technology. vol. 3, pp. 1118-1124 (2017).
10. Raut, S., Fulsunge, A.: Plant Disease Detection in Image Processing Using MATLAB. International Journal of Innovative Research in Science, Engineering and Technology. vol. 6, pp. 10373-10381(2017). https://doi.org/10.15680/IJIRSET.2017.0606034
11. Singh, V., Misra, A.: Detection of plant leaf diseases using image segmentation and soft computing techniques. Information Processing in Agriculture. vol. 4, pp. 41-49 (2017). https://doi.org/10.1016/j.inpa.2016.10.005
12. Mohanty, S., Hughes, D., Salathé, M.: Using Deep Learning for Image-Based Plant Disease Detection. Frontiers in Plant Science. vol. 7, (2016). https://doi.org/10.3389/fpls.2016.01419
13. Khirade, S. Patil, A.: Plant Disease Detection Using Image Processing. 2015 International Conference on Computing Communication Control and Automation. pp. 768-771. IEEE, Pune, India (2015). https://doi.org/10.1109/iccubea.2015.153