A Deep Convolutional Neural Network with Batch Normalization Approach for Plant Disease Detection

Fahad R. Albogamy*
*f.alhammdani@tu.edu.sa
Computer Sciences Program, Turabah University College, Taif University,
P.O. Box 11099, Taif 21944, Saudi Arabia

Summary
Plant disease is one of the issues that can create losses in the production and economy of the agricultural sector. Early detection of this disease for finding solutions and treatments is still a challenge in the sustainable agriculture field. Currently, image processing techniques and machine learning methods have been applied to detect plant diseases successfully. However, the effectiveness of these methods still needs to be improved, especially in multiclass plant diseases classification. In this paper, a convolutional neural network with a batch normalization-based deep learning approach for classifying plant diseases is used to develop an automatic diagnostic assistance system for leaf diseases. The significance of using deep learning technology is to make the system be end-to-end, automatic, accurate, less expensive, and more convenient to detect plant diseases from their leaves. For evaluating the proposed model, an experiment is conducted on a public dataset contains 20654 images with 15 plant diseases. The experimental validation results on 20% of the dataset showed that the model is able to classify the 15 plant diseases labels with 96.4% testing accuracy and 0.168 testing loss. These results confirmed the applicability and effectiveness of the proposed model for the plant disease detection task.

Key words:
Plant Disease, Deep learning, Convolutional Layer, Batch Normalization, and Agricultural Sector.

1. Introduction

Plant disease is one of the problems that can create losses in production and economy in the agricultural sector and forestry [1, 2]. Detecting such diseases in early stages can control the spreading of the diseases and increase productivity. This can achieve by using suitable management approaches, such as pesticide, fungicide and chemical for specific disease. Actually, monitoring the health status of plants is a challenge for sustainable agriculture field [3, 4].

Observing with a naked eye of the experts is the main approach used to detect diseases from plant leaf [5]. However, this perception of the human eye is not able to notice the minute difference in the affected part of the image. This is because that minute difference of the colour indicates the presence of a disease in the leaf [5]. In addition, this approach is of a high cost because it requires constant monitoring of experts in large farms [6]. Moreover, it is a waste of time, and it makes consultants too expensive, especially in some developing states and countries that farmers might have to drive long distances to contact the experts [6]. Therefore, the automatic detection of the disease on the leaves is an important topic in this field of research because it is so useful in monitoring large leaves, and by then can automatically detect the symptoms of a disease as soon as they appear on the leaves [7]. The use of computerized techniques to detect the diseases of the leaf is important because it is automatic, way less expensive, accurate, and convenient [6, 7]. In addition, computerized techniques eliminate the negative aspects of traditional methods and human errors [8]. Researches concerned with detecting plant disease using image processing and machine learning technologies began in the 80s to 90s years of the 20th century. In recent years, the use of image processing, machine, and deep learning mechanisms have been widely used in many agricultural areas to improve early detection and treatment stages [9-12]. The time factor is very important to detect the disease of the plant as quick as possible. Because almost all diseases of the plant appear on the leaves as indicators. Therefore, all researches focus on plan leaves and not the whole plant.

In this research work, a practical deep learning approach is proposed to detect plant disease automatically from its leaf image using convolutional neural network layers and batch normalization with max-pooling and dropout layers. The batch normalization process helps for standardizing the inputs to the neural network; whereas, the max-pooling layers help to reduce the size of the spatial convolved features and dropout layers can mitigate the overfitting problem in the training phase of the model.

The rest of the paper is organized as follows: Section 2 gives a brief about the related work. Section 3 gives a background and basic concepts about deep and convolutional neural networks. Section 4 explains the research methodology, including a description of the dataset used and the proposed model. The experiments and discussion with results and performance comparison are

Manuscript received September 5, 2021
Manuscript revised September 20, 2021
https://doi.org/10.22937/IJCSNS.2021.21.9.6

given in section 5. Finally, section 6 concludes and summarizes the research work.

2. Related Work

In the literature review of this study, there are many methods have been proposed in several works. For example, Geng Ying, et al in [13] discuss cucumber leaf diseases by using the image pre-processing technique. Firstly, they use the Median filter to remove noises from the image. The reason for using the median filter, when it removes the noise, it avoids the blur at the edge of the image. Then, they separate diseased leaf from the background by the threshold segmentation method. Finally, they separate diseased part from the leaf by using edge detection based segmentation and the snake model "Active Contour" to extract the contour of diseased part and to give the better result of the segmentation. The edge detection can be done by using the second-order differential operator.

Shen Weizheng, et al in [14] proposed the grading method to detect the diseases of the leaf by using image processing. Qing Yao et al. in [15] proposed the system that based on the image processing procedures to detect the diseases on the rice. They used the support vector machine (SVM) for segmentation and shape with texture for features extraction. They studied different diseases on the rice, such as "blast", "bacterial leaf blight" and "sheath blight". The detection of these diseases depends on the properties of the spots.

Nunik Kurniawati et al in [16] proposed a system which can automatically detect the diseases of paddy and correctly classify them as "Blast Disease", "Brown-Spot Disease", and "Narrow Brown-Spot Disease" by using the image processing techniques. The methodology includes different steps, such as: Acquiring images: In this step, paddy leaf images were collected. Image Segmentation and Pre-Processing: Authors used two methods for segmentation, the local entropy based threshold method with Otsu threshold method and they compared between them. Feature Extraction: In this paper, authors used shape and color features because these features can distinguish many of disease. Authors used width and height of the diseased part to conclude the types of the diseased part, as Spot, Round, Oval, Taper or Spindle. Paddy Diseases Classification: authors classified paddy diseases by using the production rule method and also the forward chaining method. The authors concluded that, the local entropy threshold method was best accuracy 94.7% of the Otsu method. Because of intensity values are different; the Otsu method cannot accurately perform the segmentation.

H. Al-Hiary et al. [17] proposed a software to detect and classify the diseases of the plant leaf automatically and they proved that by experiment. The methodology includes the use of K-means algorithm for clustering and segmentation, Co-occurrence color technique for feature extraction and neural networks (NNs) for classification of the leaf diseases that appear on the surface of the leaves. In this paper, the tested diseases are accurately detected and the types of diseases are successfully classified with a precision of around 83%.

Basvaraj Anami et al. [18] proposed the system for recognizing affected and normal agriculture produce using image processing. In this paper, authors used multi-layered back-propagation neural networks (BPNNs) as a classifier for automatic disease detection and a combination of color and texture features is also used to classify and recognize different horticulture/ agriculture produce. At the end of the research, authors concluded that the suggested system can successfully support in recognizing normal and affected produce.

Piyush Chaudhary, et al. [19] proposed method for detecting the disease spot on the plant leaf based on the color transform of the image. They compared the effect of the three color space models on the disease spots detection process. In this work, the "CIELAB", "HSI" and "YCbCr" color models were used to detect plant disease spots. The methodology includes that each image is converted from the RGB color model into the one of the above color models. Then the median filter is used for removing unnecessary spots. After that, algorithm for disease spot segmentation is used. In this paper the Otsu threshold is used on the RGB image and one component of the other color models. They applied on the “A‟, “H‟ and “Cr‟ components of the CIELAB, HIS and YCbCr color models, respectively. Experiments worked on a noisy background, the disturbance of the vein color and difference in spots color of the disease. The results demonstrated that “A‟ component of the CIELAB color model accurately detected the spots of diseases. The kind of the background, the kind of leaf and the kind of the disease have no influence on the result of this method.

In addition, Kamaljot Kailey and Gurjinder Sahdra in [20] discussed a method for detect the plant disease using edge detection, image color, and matching of the histograms. This research identifies the disease based on image content for retrieval that is primarily concerned with the accurate performance of plant disease detection. The methodology that used in this research consists of two steps. In the first step: the healthy sample and the diseased sample are training. In the second step: the test sample is training. After that, the algorithm applies the comparison technique based on the histogram and edge detection technique. This algorithm compares the testing sample with the healthy sample then it compares the testing sample with the diseased sample if the testing sample is diseased. At the end, authors conclude that this algorithm is an efficient and accurate technique for automatically disease detection of plant.

Vijayakumar and Arumugam in [21] proposed a system to recognize the disease of the powdery mildew in the "Betelvine" leaves by using image processing with pattern recognition techniques. The methodology that used in this research includes a three phases; the phase of normal leaves, the phase of fully infected leaves and the phase of the test leaves. In each phase, the mean value for each RGB color components is calculated. Then algorithm can distinguish between the tested leaves if they are normal or diseased with the powdery mildew. Authors conclude that the proposed approach is cost effective and nondestructive for detecting the disease. It only requires the digital photograph of leaf samples. This method can detect the powdery mildew disease before it extends to the whole crop.

Tushar H. Jaware et al. [22] discussed crop disease detection using image segmentation. By using the noise data filter, authors in this paper improved K-means segmentation algorithm. The methodology that used in this research, as follow: the improved K-means algorithm eliminates the noise of the dataset in the preprocessing stage. Then the improved K-means algorithm performs the clustering of the dataset. This step gives perfect clustering results. The experiment results prove that the proposed algorithm is an efficient algorithm with high clustering accuracy for the detection of plant leaves diseases.

Recently, there are different deep learning-based systems and techniques have been proposed to detect plant illness from plan leaves. Prajwal TM et al. [23] developed a LeNet-based classification system to detect tomato diseases from tomato leaf images. This research aims to utilize the CNN for automated extracting the features from plan leaf and classifying the input plant image into numerous classes of diseases rather than healthy plan. This work has attained an average accuracy about 94% proving the applicability of the neural networks-based approaches even under unstable conditions of captured images.

Atabay [24] introduced a work based on a deep residual learning technique. This work has applied the CNN for classifying the recognizable impacts of tomato plant diseases using the plant leaves images. The outcome of this study showed that the proposed pre-trained models can exploit the time for re-training rather than the required time to train the models from scratch. The limitation of those two previous studies is that they used a dataset of only one type of plant (tomato plant), which is easy task to do versus multiple types of plant with multiple diseases.

Malvika Ranjan et al. [25], presented a visual procedure that extracts the HSV (hue, saturation, value of lightness) features from the color segmented images and then trains the artificial neural network (ANN) model on them to detect healthy and sick plant. The accuracy result of this work was 80%. However, the results were still contained some errors in detection plant leaf disease.

Kawasaki et al. [26] proposed an approach using a CNN model to classify the nutritious cucumbers from the diseased cucumbers using their leaves images. They performed the work to detect two risky viral infections: ZYMV (zucchini yellow mosaic virus) and MYSV (melon yellow spot virus). They conducted the experiments on a data set contained 800 images of cucumbers leaves (300 with MYSV, 200 with ZYMV, and 300 with healthy leaves).

Jasim and AL-Tuwaijari [27] presented a system to classify and detect plant leaf diseases using a convolutional neural network (CNN) deep learning method. Agarwal et al. [28] developed a deep learning model for tomato leaf disease detection, called ToLeD. It used a CNN architecture to classify ten classes of diseases from tomato leaf images with an accuracy of 91.2%.

3. Basic Concepts of Used Methods

3.1. Artificial Neural Networks (ANNs)

Artificial neural networks (ANNs) are artificial intelligence structures that are stimulated to be like the biological human brain neural networks [29, 30]. Such networks structures can learn and acquire the knowledge by examples to perform the tasks without any specific task programming rules [29]. Basically, the ANN model consists of three kinds of specific layers, which are the input layer, hidden layer(s), and output layer as displayed in Figure 1. When the ANNs contain more than two hidden layers, it is called DNNs (Deep Neural Networks) [30, 31].

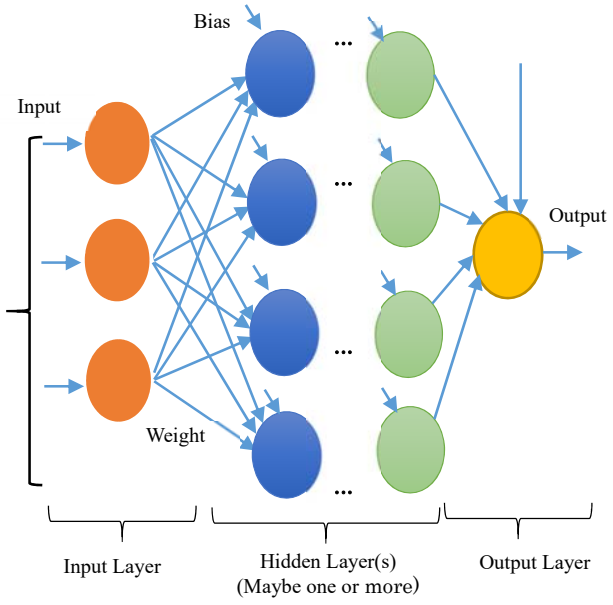


Fig. 1: Artificial neural network structure.

Individually, each layer in the ANNs structure has a number of connected units, termed nodes or neurons. These neurons incorporate the data features of the input layer and manipulate them in their layers. Each neuron can produce its output by computing the activation function for the weighted sum of previous connected neurons' inputs and weights. The weights of neurons are attuned based on the learning process administrated by the learning rule [30, 31]. The ANN model can define a mathematical model of a function with form, $f: X \to Y$, which can minimize another function, so-called a cost function calculated as follows:

$$C = \frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2$$

, where $y_i \in Y$ and $x_i \in X$ are pairs of data that can follow some kind of distribution, D.

Arithmetically, a neuron with a label j that receive the input $x_i(t)$ can contain the components as follows:

- The activation function $a_j(t)$, in which the state of the neuron depends on the parameter of discrete-time,
- The threshold θ_j that is a fixed certain value for a response, and it could be altered over the learning task,
- The activation function f, which is applied to compute a new activation function at a given time t+1 from $\theta_j, a_j(t)$ and the input $x_i(t)$ by the relation given as follows: $a_j(t+1) = f(a_j(t), x_i(t), \theta_j)$,
- and the output function f_{out}, which can compute the output result from the activation function $a_j(t)$ as follows:
- $o_j(t+1) = f_{out}(a_j(t))$,

Normally, the neurons in the input layer can work as an input interface and they have no predecessors; whereas, the neurons in the output layer can work as the output interface and they have no successors for the entire ANN model [31]. The connection between a neuron i and a neuron j can simply assign a bias b_{ij} and a weight w_{ij}. This is utilized to make a shift for the activation function. To compute the function of propagation value for the input of any neuron i from the output of any predecessor neuron j, the following equation can be calculated as:

$$\sum_{j}^{x_i} o_j(t)w_{ji} \quad (1)$$

When the bias value is added to the function of propagation value, the equation above will be changed to the following equation.

$$\sum_{j}^{x_i} o_j(t)w_{ji} + b_{ji} \quad (2)$$

The parameters of the ANNs can be changed using the learning rule of the learning task to provide with a preferred output. Learning task is usually used to adjust the outputs of the variables within the network, such as bias and weight variables [31].

3.2. Convolutional Neural Network (CNN)

The traditional ANN and DNN connect different layers of neurons as a "full connection", in which the input of each neuron in a layer is the output of one neuron in the previous layer. This connection style of the neural network is called a "Fully Connected Neural Network" [32]. It has a disadvantage because of the large number of parameters such as weights and biases. Therefore, the convergence of network model training is very slow, especially for training data such as images that have millions of pixels at every turn. Although it can converge in theory, it may have N years to produce the result, and its generalization will become worse [32].

For example, a black and white 28×28 digital image contains an object of 10 classes will have 784 neurons in the input layer, as shown in Figure 2.

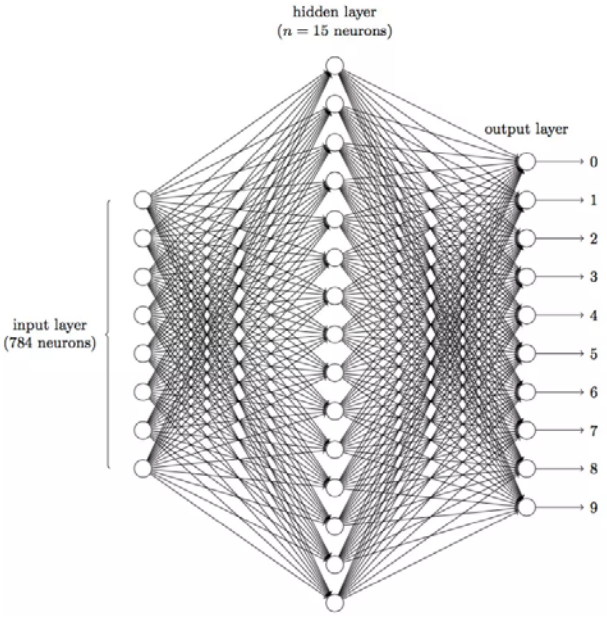


Fig. 2: Artificial neural network structure.

If only one hidden-layer is utilized in the middle and has 15 neurons, then the weight parameter (w) will have more than 784×15=11760; if the input is a 28×28 color digital image with RGB format. There are 28×28×3=2352 input neurons. It is easy to see that using a fully connected

neural network to process images requires too many training parameters. In convolutional neural networks (CNNs), the neurons in the convolutional layers are only linked to some neurons in the previous layers [32]. That is, the connections between its neurons are not fully connected, and in the same layer, the weight parameter (w) and offset b of some connections between some neurons are only shared (i.e., the same), which greatly reduces the number of required training parameters.

The feature map output by the fully convolutional neural networks is like the compound eyes of an insect [33]. Each grid is an eye, and each eye sees different things, but the field of view of each eye is the same (that is, the receptive field of each eye). Similarly, each eye works alone, without affecting each of others. Then each eye will detect the category of the object and see the characteristics and attributes of this object [33].

3.2.1 Features of Convolutional Neural Networks

Local perception

To explain this feature, for example, if an image is used as input for object recognition, it is actually not required to let every neuron receive the information of the entire image [32]. However, it is required to let the neurons in some different areas correspond to some different parts of the entire image receive the information about that parts, and finally the local information is only needed to be integrated together. This process is equal to performing a dimensionality reduction in the initial neurons of the input layer. The local perception is the local receptive field of the convolution kernel, which refers to the pixel area covered by the convolution kernel. Since the area covered by each convolution kernel is only a small part, it is a local feature, that is, local perception. CNN is a process from part to whole (the realization of a part to whole is in the fully connected layer) [32, 33].

Weight sharing

The traditional neural network has a huge amount of parameters. For example, to perform a full connection operation on a 1000X1000 pixel image, it needs (1000x1000), i.e., 10 with 6th power parameters. This can be said to be the most important aspect of the convolutional neural network. Therefore, the biggest problem with the fully-connected neural network model is that there are too many weight parameters. For the convolutional layer of the CNN, there are different neurons, and the weights are shared, which greatly reduces the values of the entire neural networks parameters and improves the performance of training through the entire network.

Down-sampling

Down-sampling is another important concept of the convolutional neural networks, which is also commonly referred to as pooling. For understanding this concept, it can be regarded as a "loss compression" of an image. This is because in the actual training, it is not required to extract and train all detail within the image [32]. Consequently, the pooling role is a further step for the information abstraction and feature extraction. It also reduces the amount of data processing. The most common methods are maximum pooling, minimum pooling, and average pooling. The main advantage of pooling operation is that the image resolution is reduced, making the network model is not easy to over-fit [33].

3.2.2 Hierarchical Structure of Convolutional Neural Network

The general structure of CNN contains an input layer, output layer, and some multiple hidden-layers [33]. The hidden-layers could be divided into convolutional layers, pooling layers, RELU layers, and fully connected layers, as shown in Figure 3.

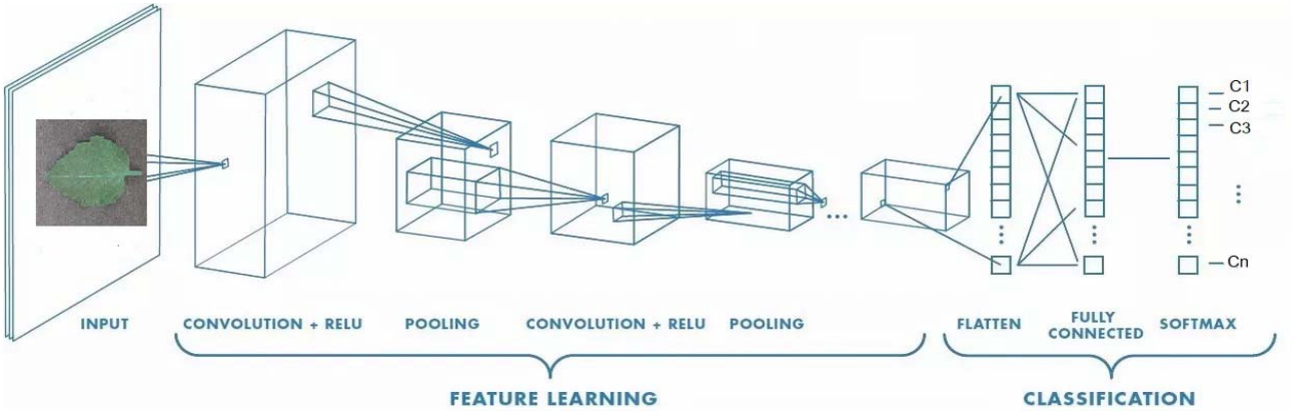


Fig. 3: Artificial neural network structure.

Input layer

The input layer is to input the original data or data preprocessed by other algorithms into the convolutional neural network. It can be a digital image or two-dimensional waveform data such as Fourier transformed in the audio recognition field. It can also be a one-dimensional sentence vector in natural language processing. The function of the input layer is to send the input data to the convolutional neural network for feature extraction, and then obtain the results. At the same time, it is possible to control the input of different amounts of data according to the difference in computing power, the difference in machine storage capacity, and the model parameters [32, 33].

In the CNN input layer, the input of one-dimensional data vector format for the fully connected neural network is not the same as the input format of data image. The input format of the input layer of CNN preserves the image structure itself. For a white and black 28×28 image, the input of CNN is a 28×28 two dimensional neurons. For 28×28 images in RGB format, the input of CNN is a 3×28×28 three dimensional neurons (each color channel in the RGB has a 28×28 matrix).

Convolutional layer

The convolutional layer is one of the most important layers in the architecture of CNN. Basically, convolution is a translation-invariant and linear operation, which consists of performing a local weighting combination on the input data [32]. Depending on the number of selected weights, different properties of the input will be revealed. Also, in the frequency domain, the modulation function is linked with the function of point spread that defines how the input frequency component is modulated by phase-shifting and scaling. Hence, selecting the suitable kernel is critical to get the most substantial and significant information contained in the input data signal, allowing the model to create better inferences about the content of the data signal [32].

Active layer

The activation layer is responsible for activating the special diagnosis extracted by the convolutional layer. Since the convolution operation is to perform the corresponding linear transformation on the input image and the convolution kernel, it is necessary to introduce an activation layer (non-linear function) to make it non-linear mapping. The activation layer is composed of nonlinear functions, such as $sigmoid$, $relu$ and $tanh$. The most commonly used activation function is $relu$, also called linear rectifier [32].

Pooling layer

If the pooling field view or filter is comparatively small and the stride is also quite small when the input data passes through the convolutional layer, the resulted feature map is still reasonably large [33]. Also, the reduction operation of high dimensionality size for each feature map can be performed by the pooling layer. Furthermore, the pooling layer has a "pooling field view (or filter)" to scan the matrix of feature map and to calculate the matrix values in the "pooling field view". Consequently, there are two calculation methods:

Max pooling: It takes the maximum value in the "pooling view" matrix.

Average pooling: It takes the average value in the "pooling view" matrix. The scanning step stride is also involved in the scanning process [33]. The process of scanning is the same as the process of convolutional layer. Scan from left to right first, then move the cloth length down when finished, and then from left to right. As shown in the example below:

Among them, "pool view" filter: 2×2; stride: 2. finally, three 24×24 feature maps can be down-sampled to obtain three 24×24 feature matrices:

In summary, the advantages of the pooling layer are:

- ***Immutability:*** It pays more attention to whether there are certain characteristics rather than the specific location of the characteristics. It can be seen as adding a strong prior, so that the learned characteristics can tolerate some changes.
- ***Reducing the input size of the next layer:*** It reduces the amount of calculation and the amount of parameters.
- ***Obtaining a fixed-length of the output.*** For example, in text classification, the input can be a variable length, and a fixed-length of the output can be achieved by pooling that also could prevent from the over-fitting or possible under-fitting.

Fully connected layer and output layer

The fully connected layer is a process of linear feature mapping, which maps the input of multi-dimensional features to the output of two-dimensional features [32]. A high-dimensional input represents the sample batch, and the low-level often corresponds to the task objective (for example, classification corresponds to each category Probability). The fully connected layer primarily refits the features to reduce the loss of feature information; the output layer is mainly prepared to output the final target result [33].

Normalization layer

Due to the presence of cascaded non-linear operations in the network, the multi-layer architecture is highly non-linear. Besides to non-linearity of rectification, normalization is another non-linear processing component that can play an important role in the architecture of CNN [33].

Batch Normalization

Batch normalization realizes the preprocessing operation in the middle of the network model layers. Specifically, the input of the previous layer is normalized and then enters the subsequent layer of the network, which can successfully avoid "gradient dispersion", to accelerate the training process of the network [32, 33].

During each training, samples of batch-size are taken for training. In the batch normalization layer, a neuron is regarded as a feature. The batch-size samples will have batch-size values in a certain feature dimension, and then the value of batch-size in each neuron dimension. The mean and variance of these samples are obtained by formulas, and then the corresponding output of each neuron is obtained by parameter and linear mapping [33].

4. Proposed Methodology

This section describes the methodology used in this research study. The aims of this methodology is to build an effective deep learning model for detecting plant disease from plant leaves. The model is trained on a large set of images contains plant leaves and utilized to classify plant diseases from healthy ones based on the leaves inputs images. In the following subsection, a description about the dataset used in the experiment and the proposed model will be explained in more detail.

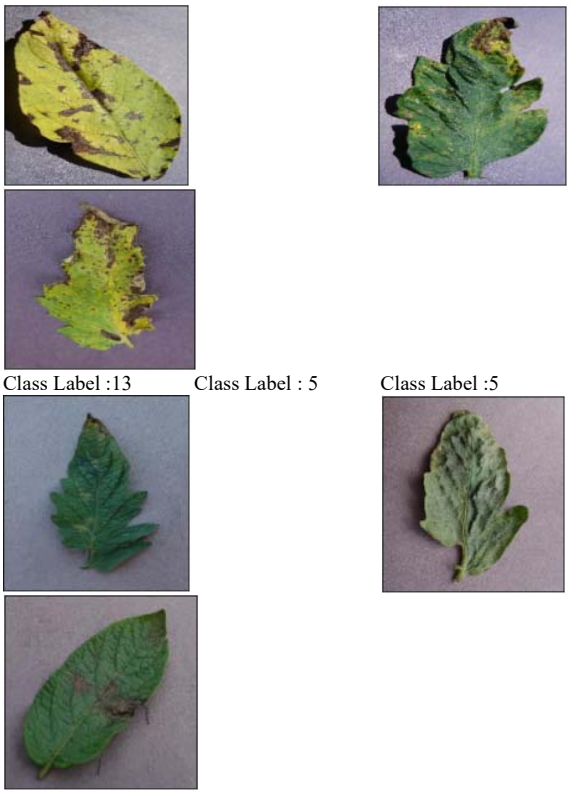
4.1. Plant Village Dataset

The benchmarked data sample adopted in this study is a Plant Village dataset. It is collected from website (www.plantvillage.org). The dataset contains around 20636 images of plan leaves for 15 classes.

These classes are related to the types of plant disease as well as the healthy types for three kinds of plant. The three kinds of the plant are pepper, tomato, and potato crops. Each image of the dataset is in RGB color space and saved in uncompressed JPG format.

Figure 4 visualizes some examples of plant leaves images taken from the datasets. Table 1 presents the distribution of images for each class with its class label.

Class Label :12 Class Label : 7 Class Label :10



Class Label :13 Class Label : 5 Class Label :5

Class Label :2 Class Label : 9 Class Label :11

Fig. 4: Visualization of some plant leaves images taken from the datasets.

Table 1: Distribution of images for each class in the Plant Village dataset.

Class Name	***Class Label***	***Number of Images***
PepperbellBacterial-spot	1	997
Pepperbell-healthy	2	1478
PotatoEarly-blight	3	1000
Potato-healthy	4	152
PotatoLate-blight	5	1000
TomatoTarget-Spot	6	2127
TomatoTomato-mosaic-virus	7	1000
TomatoTomato-YellowLeaf-Curl-Virus	8	1591
TomatoBacterial-spot	9	1909
TomatoEarly-blight	10	952
Tomato-healthy	11	1771
TomatoLate-blight	12	1676
TomatoLeaf-Mold	13	1404
TomatoSeptoria-leaf-spot	14	373
TomatoSpider-mites-Two-spotted-spider-mite	15	3209

4.2. Proposed Deep Learning Model

This section describes the proposed model. It is built based on a deep two-dimensional-convolutional neural network (2D-CNN) that contains a set of convolutional, batch normalization, max pooling, dropout, and dense layers. The model is developed to detect plant disease from plan leaves image capturing by wearable or unwearable cameras. A CNN is a kind of artificial neural network (ANNs) that has revealed representative performance on numerous competitions and applications related to image processing and computer vision for detection the objects, image classification, segmentation, and video processing.

The main advantage of CNN is the learning ability as a result of using several feature extraction phases, which can automatically learn the data representations among all dataset examples. The trend in use CNNs has been accelerated because of the availability of large datasets and the large improvements in the technology of hardware. Recently, different interesting architectures of deep CNN have been reported in [34].

By exploiting the advantages of CNN, a deep learning model using two-dimensional-convolutional (2D-Conv) layers with batch normalization, max pooling, and dropout has been developed. It consists of five blocks, as shown in Figure 5. The first block has three layers, which are a 2D-Conv layer with 32 filters, kernel size 3 by 3, and relu activation function, a batch normalization layer with axis equals one, and a 2D-Max pooling layer with pool size equals 2 by 2. Similarly, blocks 2-4 have the same layers as block 1 but with a different number of filters for 2D-Conv layers. Blocks 2 and 3 have 2D-Conv layers with 64 filters, and block 4 has a 2D-Conv layer with 32 filters. Block 5 contains flatten and three dense layers. The first and second dense layers have 512 neurons with RELU activation function, and they are followed up by a dropout layer with a drop ratio equals to 0.2. The last dense layer has 15 neurons with a SOFTMAX function. These 15 neurons are equal to the number of classes. The model takes the plant leaf image as input and outputs a label from plant disease classes or a label of healthy class. The values of model's hyper-parameters will be initialized during the experiment phase by trying different values at different runs.

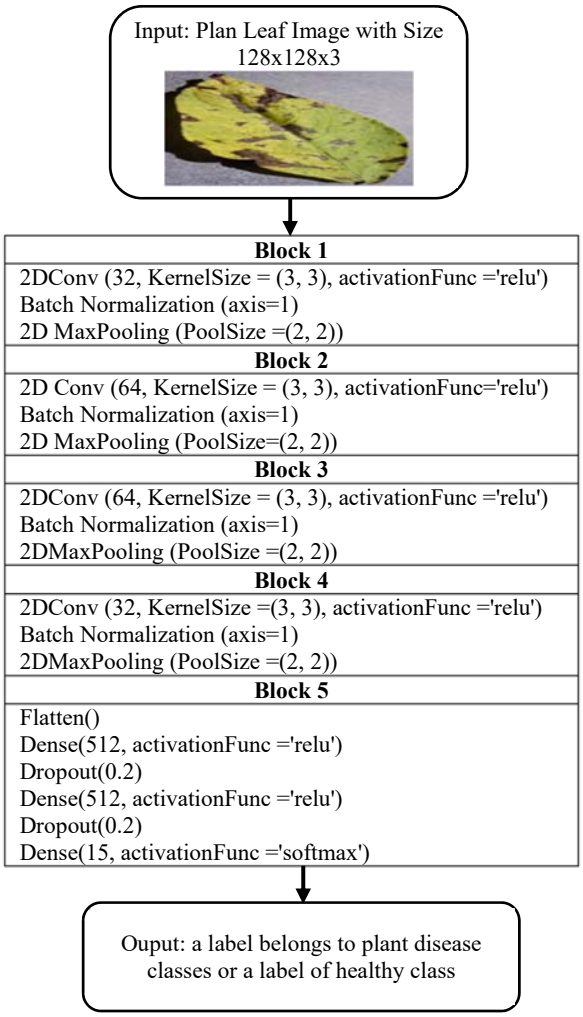


Fig. 5: The flowchart of the proposed model architecture.

5. Experiments and Discussion

5.1 Implementation Tool Description

This work was implemented using Python programming language on a laptop Intel i7 with NVidia GEFORCE GTX display card, 32GB RAM, and Windows 10 operating system. The open-source KERAS software library is a popular tool used to provide deep learning and artificial neural networks functions.

5.2 Evaluation Measures

The evaluation measures used to evaluate the proposed model performance are four metrics, given and computed by the following equations.

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)} \quad (3)$$

$$Precision = \frac{TP}{TP + FP} \quad (4)$$

$$Recall = \frac{TP}{TP+FN} \quad (5)$$

$$F1\text{-}score = \frac{2 \times (Recall \times Precision)}{(Recall + Precision)} \quad (6)$$

Where TN and TP are true negatives and true positives, FN and FP are false negatives and false positives.

5.3 Results and Comparisons

The results of the experiment are obtained using a hold-out test strategy. In this strategy, the dataset is divided into two parts: one part for training and the other part for testing. Here, in this study experiment, the dataset is split into 80% (16,511 images) to train the model and 20% (4,128 images) for evaluating its performance. For assigning the best value of model's learning rate parameter, a number of experiments are conducted using different values of learning rate. These values are in the range between 0.0001 and 0.0011. For other model's parameters, suitable values are also chosen based on the experience and reading knowledge in the field of deep learning for similar applications. Consequently, the input image size is initialized to be 128x128, Adam optimizer is used, the batch size is set with 64, and the number of the epoch is set to be 50 iterations. Figure 6 shows the accuracies of the model at the different values of the learning rate. From Figure 6, it is clear to see that the best value of model's accuracy is in the 0.0005 learning rate, in which the model achieves a 96.4% accuracy rate.

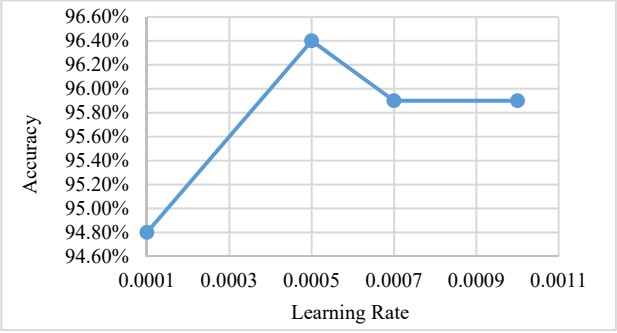
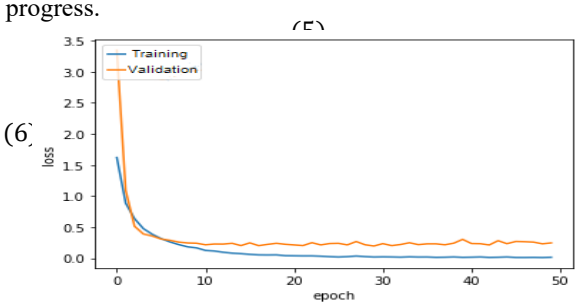
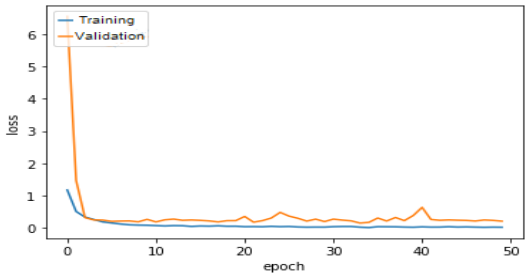


Fig. 6: Model accuracies values on the test set at different values of learning rates.

In this section, the values of loss and accuracy for model training and validation at 0.0001 and 0.0005 learning rates are shown in Figures 7 and 8, respectively. It is clear that the value of validation loss in Figure 7 (b) is almost achieved the value of 0.17, which is less than the loss value in Figure 7 (a). This means that the 0.0005 learning rate of the model is the best value. This is similar to the training and validation accuracy in Figure 8. Moreover, Figure 7 (b) and Figure 8 (b) show that the model is stable, and there is no overfitting in the training progress.

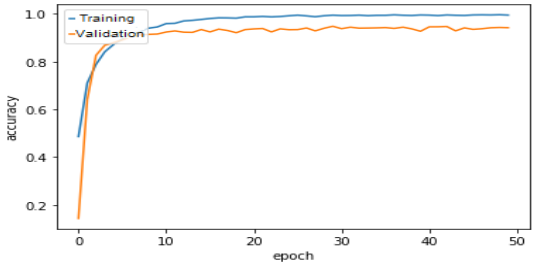


(a)

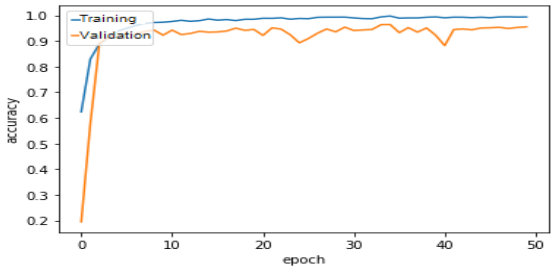


(b)

Fig. 7: Training and validation loss of the model during the training progress: (a) at learning rate 0.0001, and (b) at learning rate 0.0005.



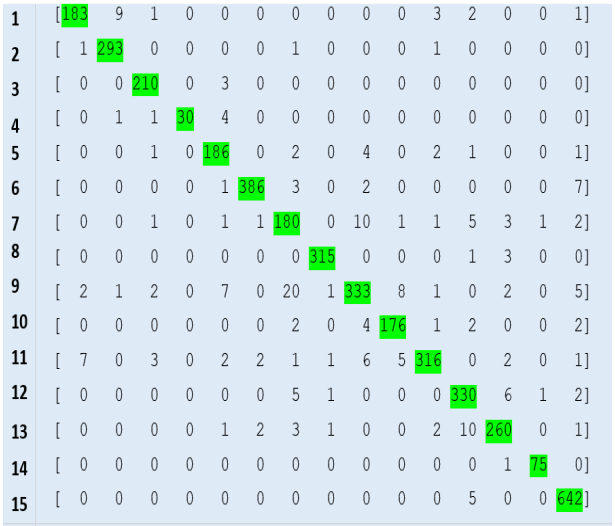
(a)



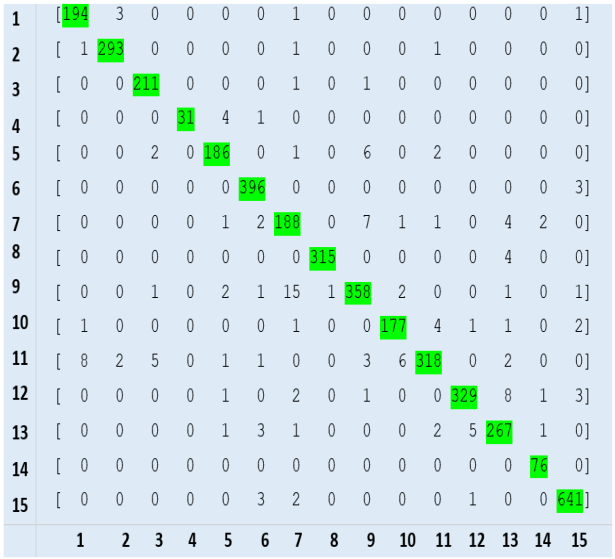
(b)

Fig. 8: Training and validation accuracy of the model during the training progress: (a) at learning rate 0.0001, and (b) at learning rate 0.0005.

The confusion matrices and quantitative results of evaluation measures for the model using 0.0001 and 0.0005 learning rates are given in Figure 9 (a) and Figure 9 (b), as well as Tables 2 and 3, respectively.



(a)



(b)

Fig. 9: Confusion matrices of correctly classified instances for the test set using different learning rate: at learning rate 0.0001, and (b) at learning rate 0.0005.

From Figure 9 (a), we can see that the model is able to classify correctly 3915 from 4128 instances when the learning rate is 0.0001, and we can notice that the model can classify correctly 3980 from 4128 instances when the learning rate is 0.0005, which is more effective.

Table 2: Experimental results of evaluation measures on the test examples using 0.0001 learning rate.

Class label	***Precision***	***Recall***	***F1-score***
1	0.948	0.920	0.934
2	0.964	0.990	0.977
3	0.959	0.986	0.972
4	1.000	0.833	0.909
5	0.907	0.944	0.925
6	0.987	0.967	0.977
7	0.829	0.874	0.851
8	0.987	0.987	0.987
9	0.928	0.872	0.899
10	0.926	0.941	0.934
11	0.966	0.913	0.939
12	0.927	0.957	0.942
13	0.939	0.929	0.934
14	0.974	0.987	0.980
15	0.967	0.992	0.979
Weighted avg.	0.949	0.948	0.948
Accuracy	94.8%		

The experimental results in Table 2 show that the proposed model achieves 94.8% accuracy, 94.9% precision, and 94.8% for both recall and F1-score at a 0.0001 learning rate. We also notice that class number 8 has a high result for precision, recall, and F1-score, which is 0.987; whereas, class number 7 achieves low results for precision, recall, and F1-score, which are 0.829, 0.874, and 0.851, respectively,

Table 3: Experimental results values of evaluation measures on test examples using 0.0005 learning rate.

Class label	***Precision***	***Recall***	***F1-score***
1	0.951	0.975	0.963
2	0.983	0.990	0.987
3	0.963	0.991	0.977
4	1.000	0.861	0.925
5	0.949	0.944	0.947
6	0.973	0.992	0.983
7	0.883	0.913	0.897
8	0.997	0.987	0.992
9	0.952	0.937	0.945
10	0.952	0.947	0.949
11	0.970	0.919	0.944
12	0.979	0.954	0.966
13	0.930	0.954	0.942
14	0.950	1.000	0.974
15	0.985	0.991	0.988
Weighted avg.	0.964	0.964	0.964
Accuracy	96.40%		

Furthermore, the proposed model with a 0.0005 learning rate attains 96.4% accuracy and 96.4% for precision, recall, and F1-score. All these results prove the effectiveness of the proposed deep learning model with a 0.0005 learning rate to accomplish a high accuracy result.

Besides the high results achieved by the study experiments, the comparative analysis of the proposed model with current related works is necessary and required. To perform this comparison, Table 4 presents and

summarizes the performance results of the developed model compared to some current works on the same dataset and according to the type of classification problem, accuracy, recall, and precision.

Table 4: Comparisons results of proposed work compared to some current related works on the same dataset.

Work	*Type of Classification Problem*	*Accuracy*	*Recall*	*Precision*
P. TM et al. [19]	10-classes	94%	94.8%	94.81%
Agarwal et al. [24]	10-classes	91%	92%	90%
This work	15-classes	96.4%	96.4%	96.4%

From Table 4, we see that the model classifier can achieve the highest accuracy result compared to related work on the same dataset for the 15-classes classification problem instead of a 10-classes classification problem. As we know, increasing the number of classes makes the decision is more complicated. Therefore, these comparisons results confirmed the applicability and effectiveness of the proposed model for the plant disease detection task.

In addition to comparison results of accuracy, recall, and precision mentioned in Table 4, a comparison analysis regarding an error rate metric is also given in Figure 10.

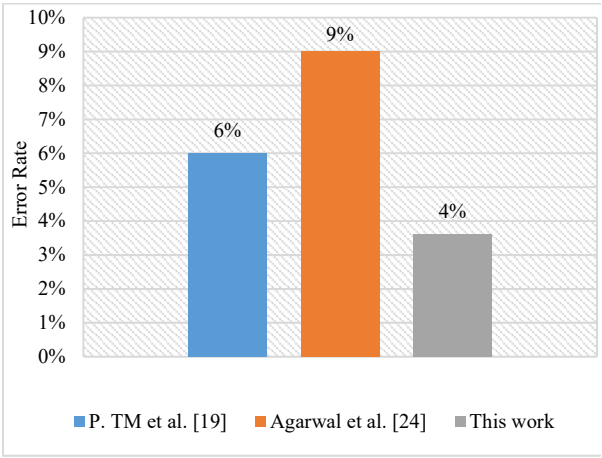


Fig. 10: Comparisons of error rate results for the proposed work against some current related works on the same dataset.

The error rate metric can be computed as the overall number of incorrectly classified examples divided by all classifications performed on the test dataset. In other words, the error rate metric is calculated by subtracting the accuracy value from one or subtracting the percentage of accuracy from one hundred percent. The low value of error rates means the best performance result in terms of corrected classification examples. From Figure 10, it is clear that the developed model also achieves the lowest value of error rates (4%) compared to the other related works. This comparison of error rates results confirms that the presented model in this study works better and more effectively than models proposed in associated compared studies.

6. Conclusion and Future Work

This paper presents an effective research methodology to detect plant disease using the images captured from plan leaves. In this methodology, a deep neural network is developed using a number of convolutional layers with batch normalization, max pooling, and dropout layers. The main contribution of the work is to solve the standardization problem of the inputs plant leaf features to the neural network by using the batch normalization process. As well, the proposed model can reduce the size of the spatial convolved plant leaf features by using the max-pooling layers and can mitigate the overfitting problem in the training phase by utilizing the dropout layers.

To validate the proposed model, an experiment is conducted on a public dataset contains 20654 images with 15 plant diseases. Also, a hold-out test strategy is used for evaluating the model, in which the dataset has divided into 80% for training and 20% for validation. The experimental results showed that the model is able to classify the 15 plant diseases labels with 96.4% testing accuracy and 0.168 testing loss. These results proved the applicability and effectiveness of the proposed model for plant disease detection tasks.

In future work, data augmentation with the proposed model will be investigated to improve more the performance of the detection task. Moreover, more datasets will be used, and some optimization techniques will be applied to select the best values of model parameters.

References

[1] Martinelli, F., Scalenghe, R., Davino, S., Panno, S., Scuderi, G., Ruisi, P. & Dandekar, A. M. (2015), "Advanced methods of plant disease detection. A review", Agronomy for Sustainable Development, Vol. 35, No. 1, pp. 1-25.

[2] Khirade, S. D., & Patil, A. B. (2015, February), "Plant disease detection using image processing", in 2015 International conference on computing communication control and automation (pp. 768-771). IEEE.

[3] Liew, O. W., Chong, P. C. J., Li, B., & Asundi, A. K. (2008), "Signature optical cues: emerging technologies for monitoring plant health", Sensors, Vol. 8, No. 5, pp. 3205-3239.

[4] Mahlein, A. K., Oerke, E. C., Steiner, U., & Dehne, H. W. (2012), "Recent advances in sensing plant diseases for precision crop protection", European Journal of Plant Pathology, Vol. 133, No. 1, pp. 197-209.

[5] Al Bashish, D., Braik, M., & Bani-Ahmad, S. (2010, December), "A framework for detection and classification of plant leaf and stem diseases", in 2010 international conference on signal and image processing (pp. 113-118). IEEE.
[6] Iqbal, Z., Khan, M. A., Sharif, M., Shah, J. H., ur Rehman, M. H., & Javed, K. (2018), "An automated detection and classification of citrus plant diseases using image processing techniques: A review", Computers and electronics in agriculture, Vol. 153, pp. 12-32.
[7] Al-Hiary, H., Bani-Ahmad, S., Reyalat, M., Braik, M., & Alrahamneh, Z. (2011), "Fast and accurate detection and classification of plant diseases", International Journal of Computer Applications, Vol. 17, No. 1, pp. 31-38.
[8] Sankaran, S., Mishra, A., Ehsani, R., & Davis, C. (2010). A review of advanced techniques for detecting plant diseases. Computers and electronics in agriculture, Vol. 72, No. 1, pp. 1-13.
[9] Agarwal, A., Sarkar, A., & Dubey, A. K. (2019), "Computer vision-based fruit disease detection and classification", In Smart Innovations in Communication and Computational Sciences, pp. 105-115. Springer, Singapore.
[10] Sethy, P. K., Barpanda, N. K., Rath, A. K., & Behera, S. K. (2020), "Image Processing Techniques for Diagnosing Rice Plant Disease: A Survey", Procedia Computer Science, Vol. 167, pp. 516-530.
[11] Reddy, J. N., Vinod, K., & Ajai, A. R. (2019, February), "Analysis of classification algorithms for plant leaf disease detection", In 2019 IEEE international conference on electrical, computer and communication technologies (ICECCT), pp. 1-6, IEEE.
[12] Rumpf, T., Mahlein, A. K., Steiner, U., Oerke, E. C., Dehne, H. W., & Plümer, L. (2010), "Early detection and classification of plant diseases with support vector machines based on hyperspectral reflectance", Computers and electronics in agriculture, Vol. 74, No.1, pp. 91-99.
[13] Ying, G., Miao, L., Yuan, Y., & Zelin, H. (2009, January), "A study on the method of image pre-processing for recognition of crop diseases", In 2009 International Conference on Advanced Computer Control, pp. 202-206, IEEE.
[14] Zhanliang, C., Changli, Z., Weizheng, S., & Xiaoxia, C. (2008), "Grading Method of Leaf Spot Disease Based on Image Processing", Journal of Agricultural Mechanization Research, Vol. 11, pp. 73-75.
[15] Yao, Q., Guan, Z., Zhou, Y., Tang, J., Hu, Y., & Yang, B. (2009, May), "Application of support vector machine for detecting rice diseases using shape and color texture features", in 2009 international conference on engineering computation, pp. 79-83, IEEE.
[16] Kurniawati, N. N., Abdullah, S. N. H. S., Abdullah, S., & Abdullah, S. (2009, December), "Investigation on image processing techniques for diagnosing paddy diseases", in 2009 international conference of soft computing and pattern recognition (pp. 272-277). IEEE.
[17] Al-Hiary, H., Bani-Ahmad, S., Reyalat, M., Braik, M., & Alrahamneh, Z. (2011), "Fast and accurate detection and classification of plant diseases", International Journal of Computer Applications, Vol. 17, No. 1, pp. 31-38.
[18] Anami, B. S., Pujari, J. D., & Yakkundimath, R. (2011), "Identification and classification of normal and affected agriculture/horticulture produce based on combined color and texture feature extraction", International Journal of Computer Applications in Engineering Sciences, Vol. 1, No. 3, pp. 356-360.
[19] Chaudhary, P., Chaudhari, A. K., Cheeran, A. N., & Godara, S. (2012), "Color transform based approach for disease spot detection on plant leaf", International journal of computer science and telecommunications, Vol. 3, No. 6, pp. 65-70.
[20] Kailey, K. S., & Sahdra, G. S. (2012), "Content-Based Image Retrieval (CBIR) For Identifying Image Based Plant Disease", International Journal of Computer Technology and Applications (May 2012), pp. 1099-1104.
[21] Vijayakumar, J., & Arumugam, S. (2012), "Recognition of powdery mildew disease for betelvine plants using digital image processing", International Journal of Distributed and Parallel Systems, Vol. 3, No. 2, pp. 231.
[22] Jaware, T. H., Badgujar, R. D., & Patil, P. G. (2012), "Crop disease detection using image segmentation", World Journal of Science and Technology, Vol. 2, No. 4, pp. 190-194.
[23] TM, P., Pranathi, A., SaiAshritha, K., Chittaragi, N. B., & Koolagudi, S. G. (2018, August), "Tomato leaf disease detection using convolutional neural networks", in 2018 eleventh international conference on contemporary computing (IC3), pp. 1-5, IEEE.
[24] Atabay, H. A. (2017), "Deep Residual Learning for Tomato Plant Leaf Disease Identification", Journal of Theoretical & Applied Information Technology, Vol. 95, No. 24.
[25] Ranjan, M., Weginwar, M. R., Joshi, N., & Ingole, A. B. (2015), "Detection and classification of leaf disease using artificial neural network", International Journal of Technical Research and Applications, Vol. 3, No. 3, pp. 331-333.
[26] Kawasaki, Y., Uga, H., Kagiwada, S., & Iyatomi, H. (2015, December), "Basic study of automated diagnosis of viral plant diseases using convolutional neural networks", in International symposium on visual computing, pp. 638-645, Springer, Cham.
[27] Jasim, M. A., & AL-Tuwaijari, J. M. (2020, April), "Plant leaf diseases detection and classification using image processing and deep learning technique", in 2020 International Conference on Computer Science and Software Engineering (CSASE). pp. 259-265, IEEE.
[28] Agarwal, M., Singh, A., Arjaria, S., Sinha, A., & Gupta, S. (2020), "ToLeD: Tomato leaf disease detection using convolution neural network", Procedia Computer Science, Vol. 167, pp. 293-301.
[29] Krogh, A. (2008), "What are artificial neural networks?" Nature biotechnology, Vol. 26, no. 2, pp. 195-197.
[30] Birdi, Y., Aurora, T., & Arora, P. (2013), "Study of artificial neural networks and neural implants", International Journal on Recent and Innovation Trends in Computing and Communication, Vol. 1, No. 4, pp. 258-262.
[31] Suk, H. I. (2017), "An introduction to neural networks and deep learning", in Deep Learning for Medical Image Analysis (pp. 3-24). Academic Press.
[32] Ciaburro, G., & Venkateswaran, B. (2017), "Neural Networks with R: Smart models using CNN, RNN, deep learning, and artificial intelligence principles", Packt Publishing Ltd.
[33] Jogin, M., Madhulika, M. S., Divya, G. D., Meghana, R. K., & Apoorva, S. (2018, May), "Feature extraction using convolution neural networks (CNN) and deep learning", in 2018 3rd IEEE international conference on recent trends in electronics, information & communication technology (RTEICT) (pp. 2319-2323). IEEE.
[34] Khan, A., Sohail, A., Zahoora, U., & Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, *53*(8), 5455-5516.

Fahad R. Albogamy received the BSc degree in Information Systems with honor from King Saud University in 2003. He received the MSc and PhD degrees in Computer Sciences with distinction from Manchester University, Uk in 2010 and 2017 respectively. He was the first dean of Applied Computer Sciences College at King Saud University. He is currently an Assistant Professor of Computer Sciences at Taif University, Saudi Arabia. He works as a consultant for academic affairs at the University Vice Presidency for Academic Affairs and Development. His research interests include Artificial Intelligence, Big Data, Machine learning, NLP and Digital Image and Signal Processing.