Neural Computing and Applications (2019) 31:8887–8895
https://doi.org/10.1007/s00521-019-04228-3

ORIGINAL ARTICLE

Maize leaf disease classification using deep convolutional neural networks

Ramar Ahila Priyadharshini[1] · Selvaraj Arivazhagan[1] · Madakannu Arun[1] · Annamalai Mirnalini[1]

Received: 1 August 2018 / Accepted: 9 May 2019 / Published online: 17 May 2019

Abstract
Crop diseases are a major threat to food security. Identifying the diseases rapidly is still a difficult task in many parts of the world due to the lack of the necessary infrastructure. The accurate identification of crop diseases is highly desired in the field of agricultural information. In this study, we propose a deep convolutional neural network (CNN)-based architecture (modified LeNet) for maize leaf disease classification. The experimentation is carried out using maize leaf images from the PlantVillage dataset. The proposed CNNs are trained to identify four different classes (three diseases and one healthy class). The learned model achieves an accuracy of 97.89%. The simulation results for the classification of maize leaf disease show the potential efficiency of the proposed method.

Keywords Deep learning · CNN · Maize leaf disease · PCA whitening

1 Introduction

Maize, very popularly known as "corn," is one of the emerging crops which is very versatile even under varied climatic conditions. It is one of the important food and industrial crops of the world. It is also named as the "queen of the cereals" as it has the highest yield among the cereal crops. It is grown in almost every country except Antarctica and also under varied range of agro-climates than any other cereal crops.

Maize diseases have a critical effect in maize production. The diseases can be visible in various parts of crops such as leaf, stem or panicle. Observing the plant with naked eye and detecting the disease results in inaccurate detection of diseases. This leads to the wrong usage of pesticides which causes harmful chronic diseases on human beings due to biomagnification and also in turn leads to reduction in quality and quantity of maize production. Thus, the detection of maize diseases plays a vital role in sheltering the high quality and high yield of maize. In this work, a technique to classify the disease in maize leaf automatically is proposed.

Zhang et al. [1] have used the genetic algorithm support vector machines (SVMs) for the classification of maize leaf diseases. To recognize and classify maize leaf diseases and healthy leaf, Alehegn [2] developed a technique based on color, texture and morphological features.

Many research works make use of artificial neural networks compared to SVM when the balanced learning is absent [3]. Jafari et al. [4] have used the artificial neural network (ANN) for scaling imbibition recovery curve. Current trends have shown that deep learned neural network is a precious tool in the field of computer vision and pattern recognition. In the year 1998, LeCun et al. [5] developed the LeNet Architecture for digit recognition. For the past two decades, LeNet architecture has been used for variety of applications. Badea et al. [6] used two different kinds of network architectures, namely LeNet and Network in Network (NiN), for various applications like recognizing burn wounds from pediatric cases, art movement and facial keypoints detection. Xu et al. [7] used LeNet for 3D object

✉ Ramar Ahila Priyadharshini
rahila@mepcoeng.ac.in

Selvaraj Arivazhagan
sarivu@mepcoeng.ac.in

Madakannu Arun
m_arun@outlook.com

Annamalai Mirnalini
mirnaramesh@gmail.com

[1] Department of ECE, Mepco Schlenk Engineering College, Sivakasi, India

recognition using volumetric representation. Recently, convolutional neural networks (CNNs) are widely used for plant leaf disease classification such as tomato [8], rice [9] and cucumber leaf [10]; Ferreria et al. [11] used CNNs for weed detection in soybean crops. So far, no research works have explored the use of deep neural network for the classification of diseases in maize leaf. In this research work, we have used the LeNet architecture for classifying diseases in maize leaf. The aim of our work is to study the potential efficiency of LeNet architecture in automated disease classification for maize leaf images by varying various parameters such as kernel size and depth of the convolutional layers.

In maize, the leaf is affected by bacterial and fungal diseases. Northern leaf blight and gray leaf spot are the bacterial diseases, whereas common rust is a fungal disease. The motivation for developing the deep network model for maize leaf disease is to support the farmers to detect the maize leaf disease in an early stage by digital camera in an accurate and efficient manner. Extraction of effective features for identifying diseases in maize leaf images is a critical and challenging task. But deep networks like LeNet can automatically extract the features from the raw inputs in a systematic way. The learned features are reckoned as the superior level abstract depiction of inferior level raw healthy and unhealthy maize leaf images. Also, deep learning-based model is considered as one of the best classifications in pattern recognition tasks to improve the analytical results. So we develop a LeNet-based deep network model to classify the maize leaf diseases.

2 Proposed methodology

In this work, we present a novel maize leaf disease classification method based on LeNet architecture. The gradient-descent algorithm is used to train the LeNet deep network. A total of 3852 maize leaf images are used from the PlantVillage dataset [12] for classification. The images from the dataset are preprocessed using PCA whitening to make the features less correlated for speeding up the feature learning algorithm. Then, the dataset is randomly partitioned into training and testing subsets. The training subset is used to train the LeNet. The testing subset is then used to assess the performance of the learned model.

2.1 Preprocessing

Preprocessing is used to improve the image data by suppressing the unwilling distortions or enhancing some features of an image for further processing. Preprocessing technique used here is PCA whitening. Principal component analysis (PCA) is a dimensionality reduction algorithm that can be used to significantly speed up the feature learning algorithm. Whitening makes the features less correlated with each other by giving all the features the same variance which reduces the training period.

Let $\{y^{(1)}, y^{(2)}, y^{(3)}, \ldots, y^{(n)}\}$ be the maize leaf images from the PlantVillage dataset. First, compute the covariance matrix of y, $\sum y$ using Eq. 1.

$$\sum y = \frac{1}{m}\sum_{i=0}^{m}(y^{(i)})(y^{(i)})^{\mathrm{T}} \tag{1}$$

Next, compute eigenvectors of $\sum y$ and stack them to the columns of U as shown in Eq. 2. Here, u_1 represents the top eigenvector of $\sum y$, u_2 represents the second eigenvector and so on.

$$U = \begin{bmatrix} | & | & & | \\ u1 & u2 & \ldots & un \\ | & | & & | \end{bmatrix} \tag{2}$$

To make the input features uncorrelated with each other, compute $y_{\mathrm{rot}}^{(i)} = U^T y^{(i)}$. The covariance matrix of y_{rot} results in diagonal matrix whose diagonal elements are λ_1, $\lambda_2 \ldots \lambda_n$. Here, $\lambda_1, \lambda_2 \ldots \lambda_{\mathrm{n}}$ be the corresponding eigenvalues of eigenvector matrix U. Now PCA-whitened data are computed according to Eq. 3.

$$Y_{\text{PCA Whitening},i} = \frac{y_{\mathrm{rot},i}}{\sqrt{\lambda_i}} \tag{3}$$

The PCA-whitened images are shown in Fig. 1.

2.2 LeNet architecture

Convolutional neural networks (CNNs) are special kind of multilayer neural networks, designed to recognize visual patterns directly from pixel images with minimal preprocessing. CNNs are inspired from multilayer perceptrons, and they maintain a spatially local correlation using a local connectivity pattern between neurons of adjacent layers. That is, the inputs of hidden neurons in layer N are from a subset of neurons in layer $N - 1$, neurons that have spatially contiguous receptive fields.

The LeNet architecture is an excellent "first architecture" for convolutional neural networks. LeNet is small and easy to understand—yet large enough to provide interesting results [5]. Originally, LeNet is designed for handwritten and machine-printed character recognition. LeNet is made up of neurons with learnable weights and biases. Each neuron accepts several inputs, takes a weighted sum over them, passes it through an activation function and responds with an output. LeNet is a five-layer network consists of 2 convolutional layers and 3 fully connected layers. It is originally designed for an input

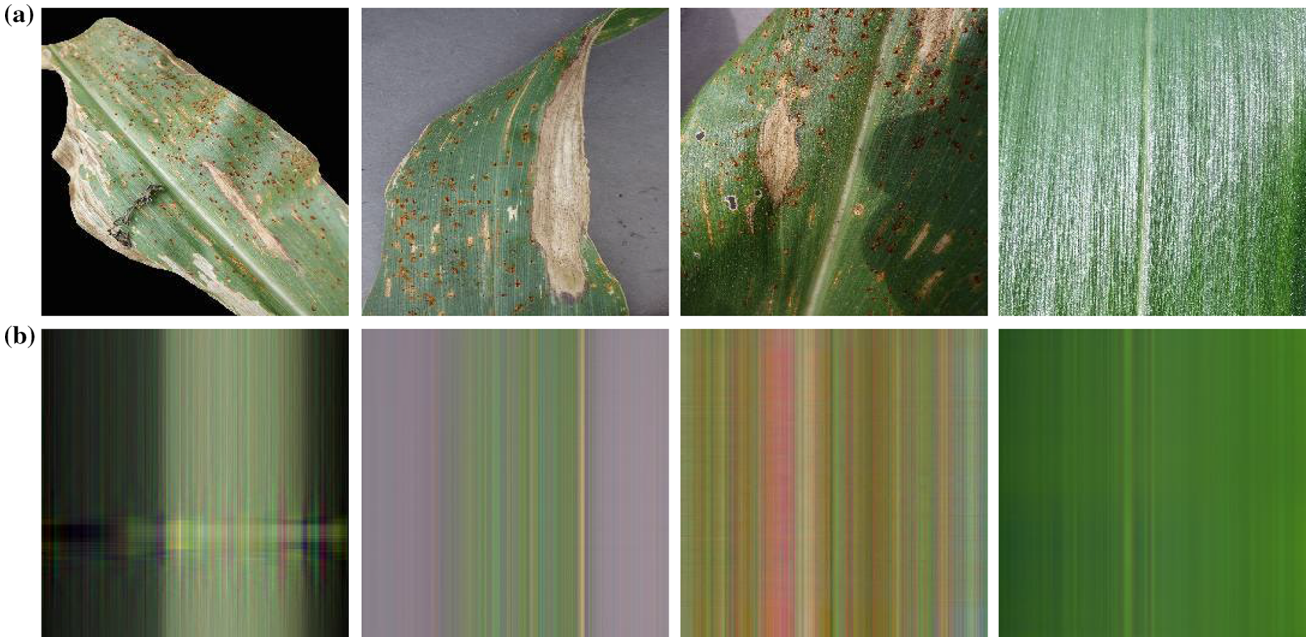

Fig. 1 **a** Sample images and **b** PCA-whitened images

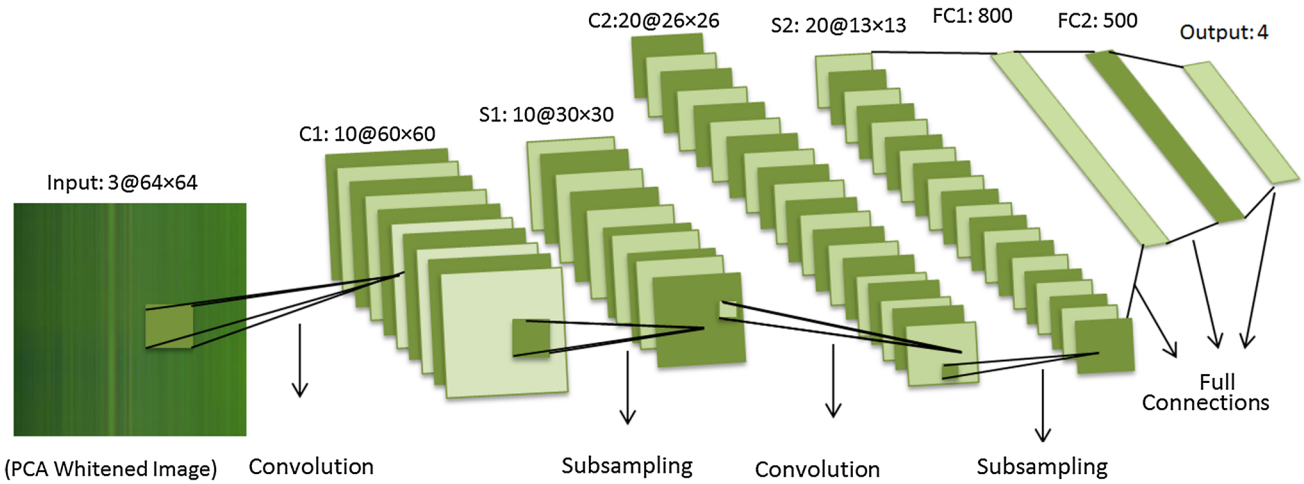

Fig. 2 Modified LeNet architecture

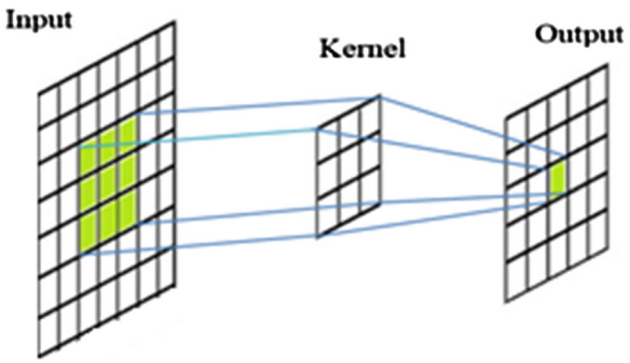

Fig. 3 Convolution operation

image of size 32 × 32. But in this work, we modified the LeNet-5 architecture to accommodate the input image of size 64 × 64. The modified LeNet architecture is shown in Fig. 2.

2.2.1 Convolutional layers

The convolution layer is the key element of a convolutional neural network [13]. The convolution layer comprises of a set of independent filters. Each filter is independently convolved with the image, and feature maps are obtained. In general, if we convolve an image of size $M \times N$ with a

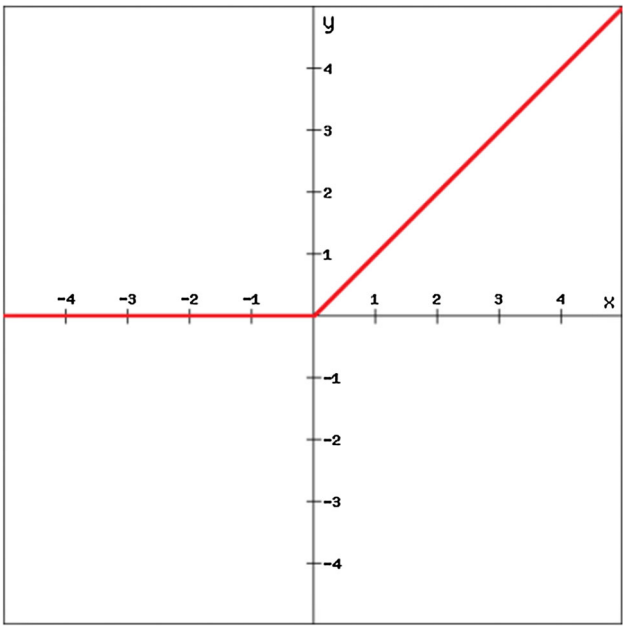

Fig. 4 ReLU activation function

filter of size $w \times h$, we get an output feature map of size $o_w \times o_h$ and it is shown in Eq. 4.

$$o_w = \frac{M - w + 2p_w}{s_w} + 1$$
$$o_h = \frac{N - h + 2p_h}{s_h} + 1 \tag{4}$$

where p_w and p_h represent the padding of zeros in width and height, respectively, and s_w and s_h represent the stride in horizontal and vertical directions. Figure 3 shows the convolutional operation of a 3×3 filter on an input map of size 7×7.

Each output feature map is obtained by the convolution of the input maps with linear filter, adding a bias term and then applying a nonlinear function. The output can be generally denoted by the formula as in Eq. 5.

$$X_j^l = f\left(\sum_{i \in I_j} X_i^{l-1} * W_{ij}^l + b_j^l\right) \tag{5}$$

where l represents the layer number, W_{ij} represents the convolutional kernel, b_j represents bias, I_j represents the set of input maps and $f(.)$ represents the activation function. The convolution layer makes the CNN to be scale invariant.

The activation function is very essential in a convolutional neural network making it capable of learning and performing more complex tasks. Activation functions are the nonlinear transformation applied to the input. They basically decide whether the information that the neuron is receiving is relevant for the given information or should it be ignored. The commonly used activation functions are sigmoid or logistic activation function, tanh or hyperbolic tangent activation function, rectified linear unit (ReLU) activation function, etc. In this work, the nonlinear activation function used is ReLU.

ReLU function is nonlinear, which means we can easily back propagate the errors and have multiple layers of neurons being activated by the ReLU function. The main advantage of using the ReLU function over other activation functions is that ReLU does not activate all the neurons at the same time. This means that, at a time, only a few neurons are activated making the network sparse, efficient and easy for computation. For negative inputs, the gradient is zero and the weights are not updated during backpropagation. This can create dead neurons which never get activated. The ReLU function is depicted in Fig. 4. The equation for the ReLU function is given in Eq. 6.

$$f(X) = \max(0, X)$$
$$f(X) = \begin{cases} X, & X \geq 0 \\ 0, & X < 0 \end{cases} \tag{6}$$

The first convolutional layer extracts different low-level features such as edges, lines and corner. Stacking of many such convolutional layers leads the network to learn more global features. So in this work, we have used two convolutional layers.

Fig. 5 Max pooling operation with a stride of 2

2.2.2 Pooling layers

Sometimes, a pooling layer is inserted in between successive convolutional layers in CNN. The function of the pooling layer is to gradually reduce the spatial size of the representation. This reduces the amount of parameters and computation in the network, and hence, it controls overfitting. Also, pooling layers make the CNN to be translation invariant. The pooling layer operates independently on every layer of the input and resizes it spatially, using the pooling operation. The most common form of a pooling layer is with filter of size 2×2 applied with a stride of 2 down-samples every depth slice in the input by 2 along both width and height, discarding 75% of the activations [13].

Spatial pooling can be of different types: max, min, average, sum, etc. For spatial pooling, spatial neighborhood of 2×2 window is defined and largest element from the feature map is taken within that window if it is max pooling as depicted in Fig. 5. Average value is taken for average pooling and so on. Max pooling gives better results for two reasons: (1) It reduces computation for upper layers by elimination non-maximal values, and (2) it provides a form of translation variance. Since it provides additional robustness to position, max pooling is a "smart" way of reducing the dimensionality of intermediate representations [14].

2.2.3 Fully connected layers

The term "fully connected" implies that every neuron in the previous layer is connected to every neuron on the current layer. Their activations can hence be computed with a matrix multiplication followed by a bias offset. The number or neurons in the last fully connected layer is same as the number of classes to be predicted. Since the default LeNet architecture is designed for digit recognition [5], the output layer is of size 10. In our work, we carried out a four-class problem, and thus, the size of the output layer is 4 as depicted in Fig. 2.

Most of the features from convolutional layers and subsampling layers are good for the classification, but the combination of those features might be even better. So the fully connected layer combines all the features extracted from the previous convolutional and subsampling layers. The final fully connected layer is using the *softmax* activation function. The softmax function is a more generalized logistic activation function which is used for multiclass classification.

2.2.4 Learning algorithm

We know that the first convolutional layer extracts low-level features such as edges, lines, curves and corner. The next level of convolutional layers learns the global features. The way the features are learnt by the CNN is through a training process called back-propagation. The back-propagation learning algorithm consists of four distinct sections, namely the forward pass, the loss function, the backward pass and the weight update.

During the forward pass, the CNN takes the training image and passes it through the whole network. Initially, all of the weights or filter values are randomly initialized. The CNN, with its current randomly initialized weights, will not be able to look for the low-level features, and thus, the CNN will not be able to make any reasonable conclusion about what the classification might be. This goes to the loss function part of back-propagation. A loss function can be defined in many different ways, but a common one is MSE (mean squared error). In this proposed work, we are using soft margin loss as defined in Eq. 7.

$$J(.) = \frac{-1}{K} \sum_{k=1}^{K} \log(1 + e^{-o_k t_k}) \tag{7}$$

where K represents the total number of classes,

$o_k \in [-1, 1]$; the output of the CNN
$t_k \in \{-1, 1\}$; target response desired.

The loss will be extremely high for the first few training images as expected. We need our CNN to predict the label which is the same as the training label. To achieve this functionality, we want to minimize the amount of the loss. Considering the minimization of the loss as an optimization problem in calculus, we wish to find out which inputs (weights in our case) most directly contribute to the loss of the network. So we need to perform a backward pass through the network, which determines the weights that contribute most to the loss and finding ways to adjust them so that the loss decreases. After the backward pass, we update the weight as shown in Eq. 8. This is the stage where the weights of the filters are updated, so that they change in the opposite direction of the gradient.

$$w^{k+1} = w^k + \eta \frac{\partial J(w)}{\partial w_k} \tag{8}$$

where w^{k+1} is the updated weight, w^k is the old weight, η is the learning rate and $\frac{\partial J(w)}{\partial w_k}$ is defined as follows.

$$\frac{\partial J(w)}{\partial w_k} = \frac{\partial J(w)}{\partial o} \cdot \frac{\partial o}{\partial Y_d} \left(\frac{\partial Z_{d-1}}{\partial Y_{d-1}} \cdot \frac{\partial Y_{d-1}}{\partial w_{d-1}} \cdots \frac{\partial Z_2}{\partial Y} \cdot \frac{\partial Y_2}{\partial w_2} \right) \cdot X$$

where $\frac{\partial J(w)}{\partial o}$ is the $\nabla J(.)$, X is the input to the CNN and the other terms are termed as $\nabla(\text{net})$.

Table 1 Details of maize leaf images in PlantVillage dataset

Class	No. of images
Common rust	1192
Gray leaf spot	513
Northern leaf blight	985
Healthy	1162

The process of forward pass, loss function, backward pass and weight update is for single iteration. The process is repeated for a fixed number of iterations or till the loss function reaches a threshold. Now, the network should have been trained well enough so that the weights of the layers are tuned correctly.

3 Experimental results and discussion

The proposed CNN model is applied to maize leaf disease recognition problem. The experimentation is carried out using PlantVillage dataset [12]. The dataset consists of four different classes. Among them, one class consists of

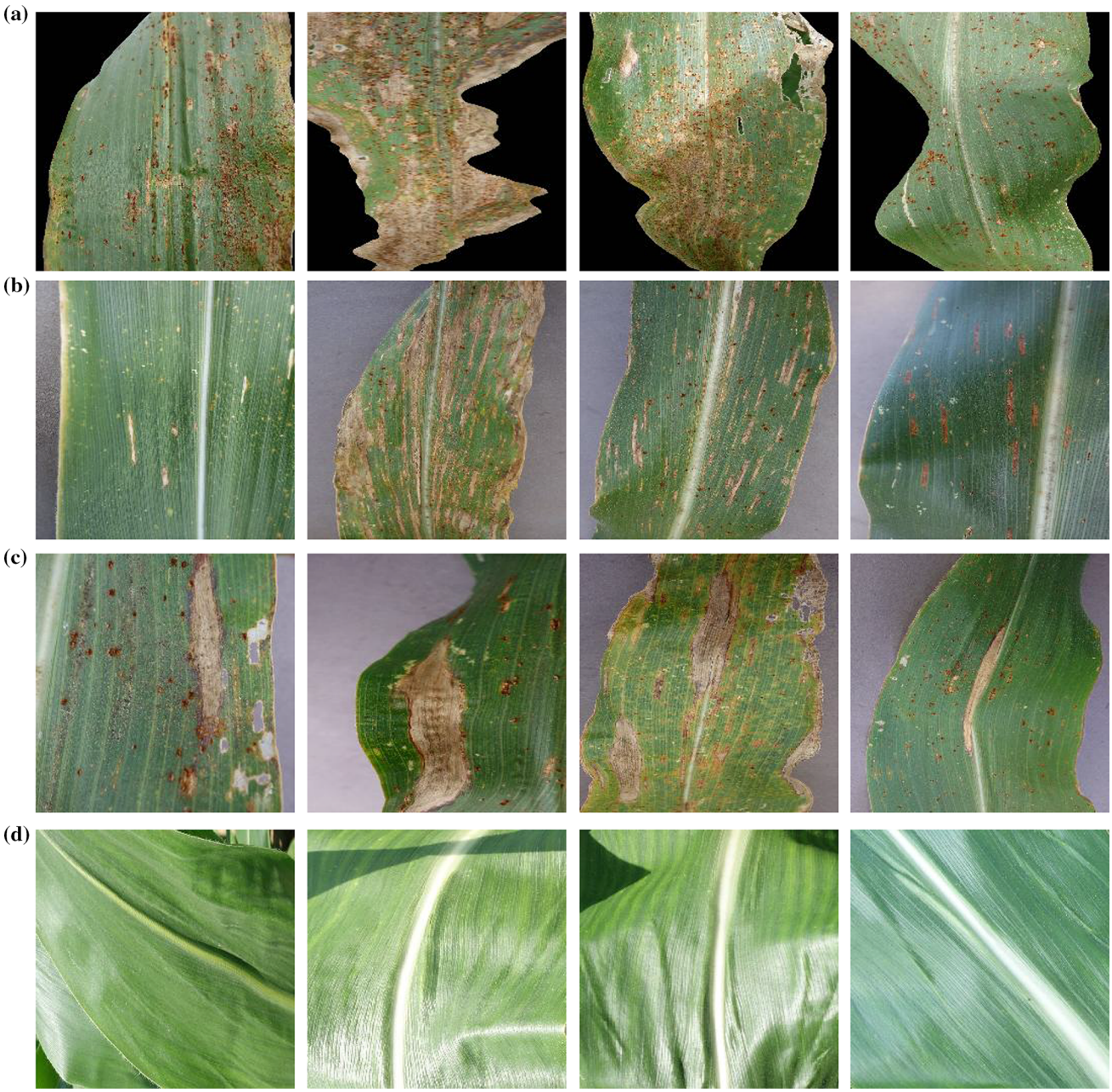

Fig. 6 Sample images from PlantVillage dataset: **a** common rust, **b** gray leaf spot, **c** northern leaf blight, **d** healthy

Table 2 Parameters of the modified LeNet architecture

Layer name	Task	Kernel size	Depth	Output dimensions
Input			3	64 × 64
C1	Convolution	5 × 5	6	60 × 60
S1	Max pooling	2 × 2	6	30 × 30
C2	Convolution	5 × 5	16	26 × 26
S2	Max pooling	2 × 2	16	13 × 13
Output	Classification			1 × 1

Table 3 Classification accuracy using modified LetNet architecture

Train–test ratio (%)	Classification accuracy (%)	
	500 epochs	1000 epochs
50–50	84.91	85.85
75–25	86.53	87.46
80–20	88.06	89.20

healthy maize leaf images and the other three classes are the common diseases in maize leaf. The maize leaf images are of size 256 × 256. The images are resized to 64 × 64 for the experimentation purpose. The details of the dataset are given in Table 1. The sample images of PlantVillage dataset are shown in Fig. 6.

Initially, the resized images are preprocessed using PCA whitening, so that the features become less correlated. The PCA-whitened images are trained using the modified LeNet architecture. The parameters of the modified LeNet architecture is shown in Table 2. The experimentation is carried out with different train and test ratios. The classification accuracy for the various train and test ratio for the above CNN network described in Table 2 is depicted in Table 3, and the classwise performance of Table 3 with 1000 epochs is shown in Table 4.

From Table 4, it is observed that, when comparing the overall performance of all four classes, the class gray leaf spot showed less accuracy which affects the overall accuracy of the maize disease classification model. This is due to class imbalance of the class gray leaf spot which has comparatively less number of images. To make the dataset as a balanced one, we have performed data augmentation using horizontal flip for the class gray leaf spot. So, this class now contains 1026 images (513 original + 513 horizontal flip). From Table 3, it is evident that the classification accuracy is high for 1000 epochs. So, further experimentation is carried out with the balanced dataset for 1000 epochs. The performance measure with the balanced dataset for 1000 epochs using different train and test ratios is depicted in Table 5.

From Table 5, it is observed that classification accuracy is improved for the balanced dataset. So far, the experimentation is done using the depth and the kernel size as mentioned in Table 1. For improving the classification accuracy, once again the proposed architecture is modified by varying the depth and kernel size. Huge hike in depth value leads to over-fitting. Thus, the depths are varied slightly. The performance measure for the balanced dataset with different kernel sizes and depths is shown in Table 6.

From Table 6, it is clear that kernel size 3 × 3 outperforms the other kernel sizes irrespective of the variation in

Table 4 Classwise classification accuracy using modified LetNet architecture

Train–test ratio (%)	Classes	Classwise classification accuracy (%)	Average classification accuracy (%)
50–50	Common rust	96.69	85.85
	Gray leaf spot	57.48	
	Northern leaf blight	90.57	
	Healthy	98.36	
75–25	Common rust	98.32	87.46
	Gray leaf spot	60.86	
	Northern leaf blight	92.01	
	Healthy	98.69	
80–20	Common rust	99.49	89.20
	Gray leaf spot	64.48	
	Northern leaf blight	93.77	
	Healthy	99.06	

Table 5 Performance measure for the balanced dataset

Train–test ratio (%)	Classes	Classwise classification accuracy (%)	Average classification accuracy (%)
50–50	Common rust	99.04	91.97
	Gray leaf spot	74.67	
	Northern leaf blight	94.99	
	Healthy	99.21	
75–25	Common rust	99.53	94.66
	Gray leaf spot	81.78	
	Northern leaf blight	97.82	
	Healthy	99.53	
80–20	Common rust	99.87	95.57
	Gray leaf spot	84.58	
	Northern leaf blight	98.14	
	Healthy	99.70	

Table 6 Performance measure for the balanced dataset with different depths and kernel sizes

Train–test ratio (%)	Classwise classification accuracy (%)					
	Depth: C1@6,C2@16			Depth: C1@10,C2@20		
	Kernel: 3 × 3	Kernel: 5 × 5	Kernel: 7 × 7	Kernel: 3 × 3	Kernel: 5 × 5	Kernel: 7 × 7
50–50	92.71	91.97	90.91	94.19	93.21	92.22
75–25	95.44	94.66	93.02	96.83	95.99	94.68
80–20	96.81	95.57	94.15	97.89	96.75	95.26

Table 7 Parameters of the proposed method

Layer name	Task	Kernel size	Depth	Output dimensions
Input			3	64 × 64
C1	Convolution	3 × 3	10	60 × 60
S1	Max pooling	2 × 2	10	30 × 30
C2	Convolution	3 × 3	20	26 × 26
S2	Max pooling	2 × 2	20	13 × 13
Output	Classification			1 × 1

Table 8 Comparison of our proposed method with other methods

Method	Classification accuracy %
GA-SVM [1]	92.82
Artificial neural network classifier [2]	94.4
Support vector machine [15]	89.6
Our proposed methodology	**97.89**

the depth. Also slight increase in the depth gives more accurate results. Highest accuracies are shown in bold. The architecture yielding highest accuracy is shown in Fig. 2, and the corresponding parameters of our proposed method are shown in Table 7.

The performance measure of our proposed method compared with those reported in the literature [1, 2, 15] is shown in Table 8. Experimental results show that the proposed methodology can effectively recognize the maize diseases.

In [1, 2, 15], the experiments are carried out with lesser number of images only and those images were collected from various sources. In our work, we have used images only from the PlantVillage dataset. We have carried out the data augmentation and worked with a total of 4365 images.

4 Conclusion

A deep convolutional neural network (CNN)-based architecture (Modified LeNet) for the classification of maize leaf disease is proposed, and its usage has been explored in detail. This method focuses on classifying the various diseases of maize leaves by learning the local and global features together. Also, in this paper we have studied the potential efficiency of the LeNet architecture for plant leaf disease classification by varying the parameters like kernel size and depth. From this study, we infer that kernel of size 3×3 is better suited for maize leaf disease classification. Further, this proposed CNN can also be used for other plant leaf disease classification.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Zhang Z, He X, Sun X, Guo L, Wang J, Wang F (2015) Image recognition of maize leaf disease based on GA-SVM. Chem Eng Trans 46:199–204
2. Alehegn E (2017) Maize leaf diseases recognition and classification based on imaging and machine learning techniques. Int J Innov Res Comput Commun Eng 5(12):1–11
3. Ren J (2012) ANN vs. SVM: which one performs better in classification of MCCs in mammogram imaging. Knowl Based Syst 26:144–153
4. Jafari I, Masihi M, Zarandi MN (2018) Scaling of counter-current imbibition recovery curves using artificial neural networks. J Geophys Eng 15(3):1062–1070
5. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
6. Badea MS, Felea II, Florea LM, Vertan C (2016) The use of deep learning in image segmentation, classification and detection. arXiv:1605.09612 [cs.CV]
7. Xu X, Dehghani A, Corrigan D, Caulfield S, Moloney D (2016), Convolutional neural network for 3D object recognition using volumetric representation. In: First international workshop on sensing, processing and learning for intelligent machines (SPLINE)
8. Brahimi M, Boukhalfa K, Moussaoui A (2016) Deep learning of tomato diseases: classification and symptoms visualization. Appl Artif Intell. https://doi.org/10.1080/08839514.2017.1315516
9. Yang L, Yi S, Zebg N, Liu Y, Zhang Y (2017) Identification of rice diseases using deep convolutional neural networks. Neurocomputing 267:378–384
10. Kawaski R, Uga H, Kagiwada S, Iyatomi H (2015) Basic study of viral plant diseases using convolutional neural networks. In: Proceedings of the international symposium on visual computing, pp 638–645
11. dos Santos Ferreria A, Freitas DM, da Silva GG (2017) Weed detection in soybean crops using ConvNets. Comput Electron Agric 143:314–324
12. https://github.com/spMohanty/PlantVillage-Dataset
13. http://cs231n.github.io/convolutional-networks/#overview
14. Li F-F, Johnson J, Yeung S (2017) Convolutional neural networks for visual recognition lecture notes
15. Zhang L, Yang B (2014) Research on recognition of maize disease based on mobile internet and support vector machine technique. Adv Mater Res 905:659–662. https://doi.org/10.4028/www.scientific.net/AMR.905.659