Indian Journal of Agricultural Sciences **91** (9): 1362–7, September 2021/Article
https://doi.org/10.56093/ijas.v91i9.116089

Image-based identification of maydis leaf blight disease of maize (*Zea mays*) using deep learning

MD ASHRAFUL HAQUE[1], SUDEEP MARWAHA[1], ALKA ARORA[1], RANJIT KUMAR PAUL[1], KARAMBIR SINGH HOODA[2], ANU SHARMA[1] and MONENDRA GROVER[1]

ICAR-Indian Agricultural Statistics Research Institute, New Delhi 110 012, India

Received: 02 January 2020; Accepted: 12 February 2021

ABSTRACT

In recent years, deep learning techniques have become very popular in the field of image recognition and classification. Image-based diagnosis of diseases in crops using deep learning techniques has become trendy in the current scientific community. In this study, a deep convolutional neural network (CNN) model has been developed to identify the images of maydis leaf bight (MLB) (*Cochliobolus heterostrophus*) disease of maize (*Zea mays* L.) crop. A total of 1547 digital images of maize leaves (596 healthy and 951 infected with maydis leaf blight disease) have been collected from different agricultural farms using hand-held camera and smartphones. The images have been collected from the experimental plots of BCKV, West Bengal and ICAR-IARI, New Delhi during 2018–19. The architectural framework of popular state-of-the network 'GoogleNet' has been used to build the deep CNN model. The developed model has been successfully trained, validated and tested on the above-mentioned dataset. The trained model has achieved an overall accuracy of 99.14% on the separate test dataset.

Keywords: Convolutional neural networks (CNNs), Deep learning, GoogleNet, Image recognition, Maize, Maydis leaf blight (MLB)

In India, maize (*Zea mays* L.) is the major crops after Rice and Wheat, supporting the overall economy of the country by providing food as well as feed. However, infestation of diseases has caused a potential loss in the overall production of the Maize crop. Maydis leaf blight (MLB), a serious fungal disease of maize becoming quite prevalent in India due to the warm and humid climate (Malik *et al.* 2018). The disease is caused by fungus *Cochliobolus heterostrophus,* which can reduce the crop yield by 40% in severe condition (Malik *et al.* 2018). The symptoms of the disease appear as small, diamond-shaped lesions at the early stages and get elongated as they mature (Singh and Srivastava, 2016). The researchers as well as the farmers commonly follow the conventional visual assessment of infected plant parts for identifying the diseases but it is error-prone, tedious and time consuming (Dechant *et al.* 2017). Hence, there is much need for automated diagnosis of symptoms of the diseases in crops.

Present address: [1]ICAR-Indian Agricultural Statistics Research Institute, Library Avenue, New Delhi; [2]ICAR-National Bureau of Plant Genetic Resources, Pusa Campus, New Delhi. *Corresponding author e-mail: ashrafulhaque664@gmail.com.

In recent years, the image recognition has gained huge popularity in agriculture sector (Kamilaris *et al.* 2018). Deep learning (DL) has brought automation in the field of disease identification using a large image dataset. The DL techniques allow the machines to dig out the valuable patterns from large dataset. The convolutional neural networks (CNNs) are the state-of-the-art framework of the deep learning techniques for tackling the image recognition problems (LeCun *et al.* 2015). Sladojevic *et al.* (2016) and Mohanty *et al.* (2016) have used different CNN models to identify images of multiple diseases of different crops. Dechant *et al.* (2017), Zhang *et al.* (2018), Marwaha *et al.* (2019) and Priyadharshini *et al.* (2019) have worked on identifying diseases of Maize crop. They have used deep learning (CNN) models to train the images of maize diseases and achieved significant results. In this experiment, a deep learning model has been developed to accurately identify the MLB disease of maize. In-field images of MLB diseases have been collected from different farm sources. The deep CNN model has been trained, validated and tested with collected images.

MATERIALS AND METHODS

Data set: It is very well known fact that the deep learning models require a huge amount of data for the image recognition task. From the large number of data, a deep learning model can easily identify important features in the

data. In this experiment, a total of 1547 images of maize leaves have been collected. All the images are captured in real in-field conditions from different agricultural farms in BCKV, West Bengal and ICAR-IARI, New Delhi during 2018–19. The images have been captured using handheld digital cameras ((Nikon D3500 W/AF)) and smartphones (Redmi Y2 and Asus Max Pro M1). There are 596 images of healthy maize leaves and 951 images of maydis leaf blight disease infected maize leaves.

Image pre-processing: Before starting the training process, pre-processing of the raw images needs to be done for accurate, effective and efficient modelling of the data (Nigam and Jain 2020). As the images have been captured using different types of instruments, there exist differences in sizes, resolutions and format of the images. The images have been normalized in terms of the sizes and format. All the images of the dataset have been reshaped to 256 × 256 sized pixels using the 'matplotlib' framework using python programming language. The reshaping of the images drastically reduces the training time of the model with very minute loss of information (Misra *et al*. 2020).

Augmentation: As we know, the deep learning model requires a lot of data for learning the patterns from the data especially image data. For better performance and to reduce the chance of overfitting of the deep learning model, there is a need for augmentation of the original dataset (Krizhevsky *et al*. 2012). The data augmentation technique is the transformation of the orientation, geometry or intensity of the original images. It includes rotation, cropping, random shifting, mirroring, zooming, altering the brightness, adding some noise, changing the contrast and so on. There are mainly two ways to augment the original dataset-first is to apply these transformations before starting the training phase and the second is to apply during the training process when each image batch is uploaded, it is also known as online data augmentation (Boulent *et al*. 2019). In this study, we have applied the online data augmentation technique using the Image Data Generator class of keras library in python. This approach applies the following transformation to each of the original images–feature standardization, random rotation, rescaling, random shift and vertical-horizontal flips. These transformations are applied only when the model is fitted with the dataset during the training phase. The main advantage of this approach is that the transformed images aren't being stored in the disk and hence no storage loss due to augmentation.

Convolutional Neural Networks: The convolutional neural networks (CNNs) have become the state-of-the-art architecture in the field of computer vision. The CNNs are the advanced version of the Artificial Neural Networks which are made up of neurons having learnable weights and biases. The CNNs have the capability to automatically learn the pattern in the images that are invariant of translation such as shifting, distortion, scaling etc. and can mine feature hierarchies (LeCun *et al.* 2010). The CNN network receives raw pixels image as inputs, performs dot products on the pixels using a set of filters then follows with non-linear function and finally generates the classification scores. The CNNs mainly consists of three components such as convolution layer, pooling layer, and dense layer.

Convolution layer: The convolution layer is the first and most important layer in the convolutional neural networks. It allows the network to learn elementary visual features (local patterns or low-level features) such as oriented edges, corners, endpoints, textures and so on from the input images (LeCun *et al.* 1998). These leaned features are then combined in the following layers for detecting the higher-order features in the images. The convolution operation extracts the features by convolving a set of filters (or kernel or weight matrices) through the image pixels and generates a set of feature maps. Each unit of a feature map is connected to several local receptive fields in the feature maps of the previous layer through the set of filters (or set of weight matrices) (LeCun *et al.* 2015). The set of units in a feature map share the same set of filters (or set of weight matrices), whose same receptive fields are located in different positions in the images. The local weighted sum of the input image pixels and the filters (weight matrices) are passed through a non-linear activation function. In general, the rectified linear unit (ReLu) is the most widely used activation function in CNNs, which performs the non-linear transformation of the input feature maps. The feature maps are calculated as (1);

$$x_j^L = f\left(\sum_{j=C_j} x_j^{L-1} . W_{ij}^L + b_j^L\right) \quad (1)$$

where, L denotes the L^{th} layer of the network, x_j^L is the output of j^{th} feature map at L^{th} layer, W_{ij}^L represents the convolution kernel of the layer, C_j is the number of input feature maps and *f(.)* is the relu activation function which is defined as (2);

$$f(x_j) = \max(0, x_j) \quad (2)$$

where, x_j is the j^{th} feature map of the network. The use of Relu in deep CNNs has made the training speed of the network faster several times than the other activation functions (Krizhevsky *et al*. 2012).

Pooling layer: The pooling layer is the next key layer of the convolutional neural networks. It is a subsampling process where the feature maps, generated from the previous convolution layer, are down-sampled. In convolutional neural network, the most commonly used pooling technique is max-pooling in which a max-filter is applied over a non-overlapping region in the feature maps. The maxpooling operation reduces the spatial resolution of the feature maps which in turn imposes the more insensitivity towards the shift or distortion variations. Therefore the pooling operation shrinks the number of learnable parameters in the network and reduces the computational load in the resources (Priyadharshini *et al.* 2019). The convolution and max-pooling operation together act as the automated feature extractor for the convolutional neural networks.

Fully connected layer (fc layer): The fully connected layer (or *fc* layer) actually resembles the architecture of the artificial neural network with an input layer, an output and

one or more hidden layers between them. In this layer, each neuron of the current layer is connected to every neuron of the next layer. The output feature maps from previous layers (convolution and max-pooling) are primarily 2D matrices and get flattened to a 1D vector of features before input to the fully connected layer. The last layer in the fully connected layer contains the same number of neurons as the total number of classes considered in the classification problem. A softmax activation function is applied in the last fully connected layer for converting the non-normalized output of the network into a probability distribution over predicted output classes.

Architecture of developed deep CNN model: In this experiment, a deep convolutional neural network model has been developed based on a popular deep learning model GoogleNet (or InceptionNet) (Szegedy *et al.* 2015) architecture for training the maize dataset. The GoogleNet model is one the most popular state-of-the-art deep CNN developed by Google team in 2014. This network architecture is very much deeper and wider, consisting of 22 layers of stacked convolutions and pooling layers. This model has 5 million network parameters which is much less as compared to the other state-of-the-art CNN architectures. In this model, fully connected layers modified to the sparse ones for dropping the number of network parameters and to avoid high computational resource consumption during training (Szegedy *et al.* 2015). The main feature of GoogleNet architecture is the 'Inception' module. The inception module achieves the integration of the sparse structure of the convolutional neural network in the present network by approximating the readily available dense components (Szegedy *et al*. 2015). The inception module consists of multiple convolution layers in parallel with size 1 × 1, 3 × 3, 5 × 5 and a max-pooling layer of 3 × 3, size, which also known as naïve inception module. Later, a convolution of 1 × 1 size has been added before the 3 × 3 and 5 × 5 convolutions to avoid the representational bottleneck problem of the inception module. In advanced versions of the GoogleNet models (*InceptionNet_v2 and Inceptionet_v3*), some more changes have been done in the inception module such as factorization of the 5 × 5 convolution into two 3 × 3 convolutions and factorization of any convolution of $n \times$ n (where $n >= 3$) size into the combination of $1 \times n$ and $n \times 1$ convolutions (Szegedy *et al.* 2016). The entire convolution in the inception module uses the rectifier linear unit (ReLu) activation function.

In this study, the deep CNN model has been achieved by applying some modifications on the base architecture of the original GoogleNet model. The modifications of original GoogleNet model includes removal of the 1000-unit final softmax layer and adding new layers such as one Global Average Pooling layer and two fully connected layers (or fc layers). This modification in the original GoogleNet model has given rise to improved deep CNN model. The GlobalAveragePooling layer flattens the feature maps from the previous layer of the network into 1D feature vectors. The first fc layer consists of 1024 units along with a rectifier linear unit (ReLu) activation function. The second fc layer contains only 2 units for classifying the learned features into two classes, i.e. healthy or MLB. And finally, a softmax activation function has been added with the second fc layers for making the prediction in the range of [0, 1]. In this experiment, a dropout layer (Srivastava *et al.* 2014) with value 0.5, has been integrated between the two *fc layers* so that the overfitting problem of the model can be reduced. The overall architecture of the developed deep CNN model has been shown in pictorial format in Fig 1. The developed deep CNN model has been optimized using the stochastic gradient descent (SGD) algorithm with a learning rate of 0.0001 and a momentum of 0.9. The learning rate has been

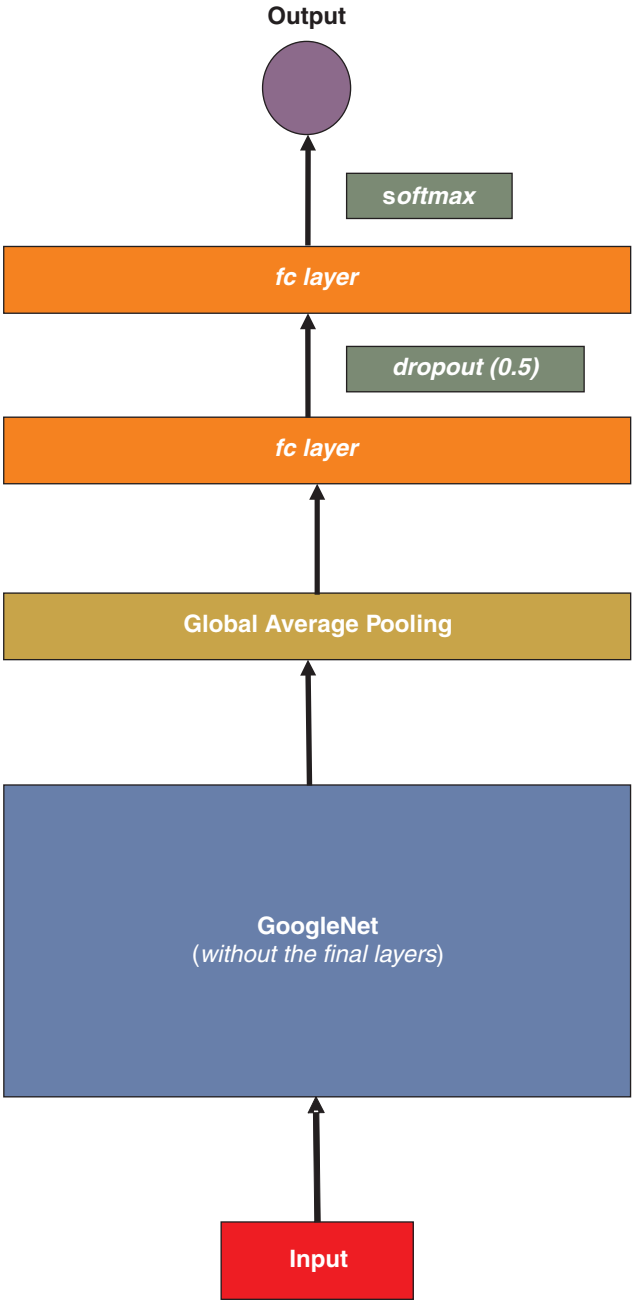

Fig 1 Overall architecture of the developed deep CNN model.

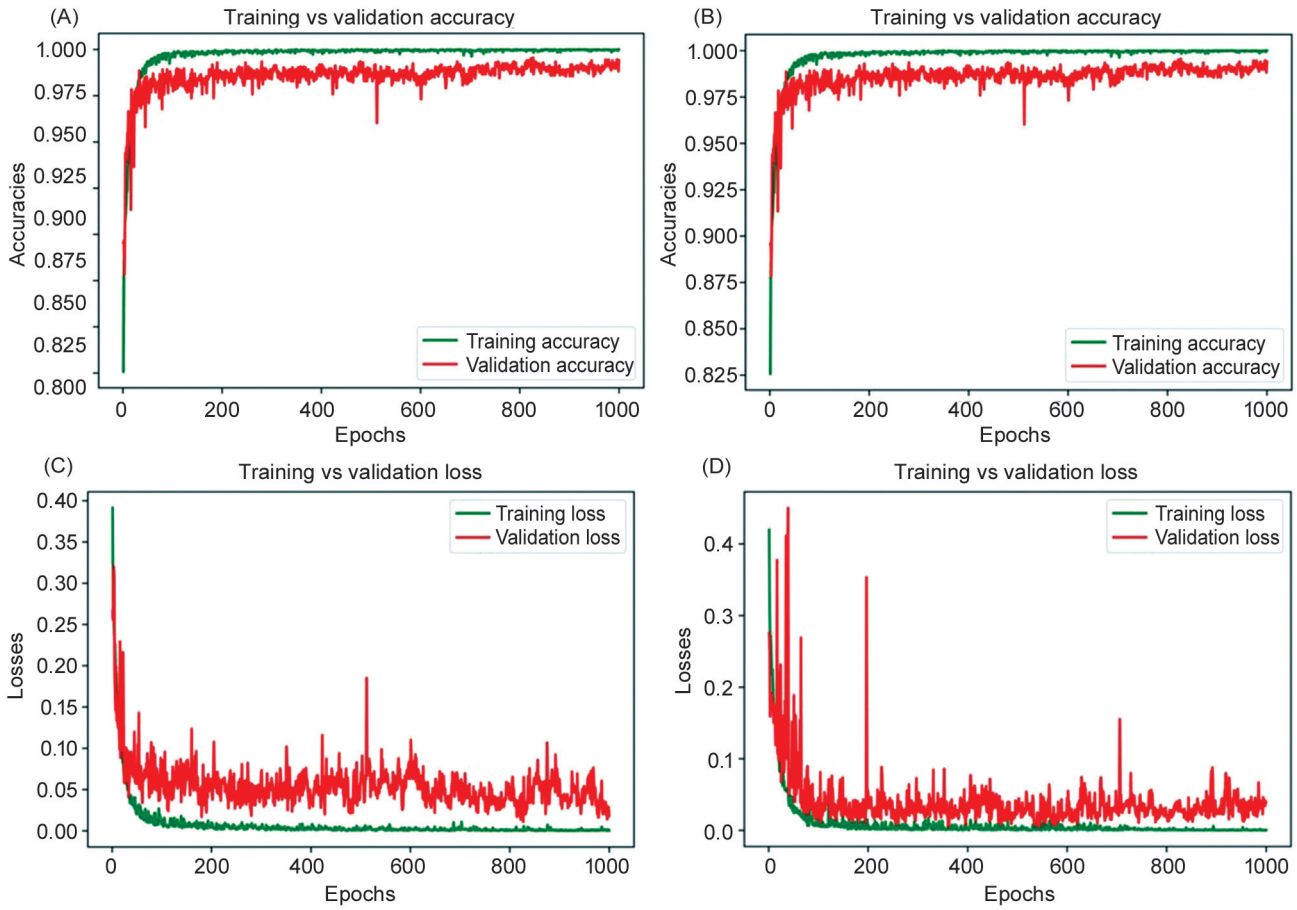

Fig 2 Trends of accuracies of the deep CNN model (A) on 70:15:15 data, (B) on 60:20:20 data and trends of losses of the deep CNN model, (C) on 70:15:15 data and (D) on 60:20:20 data.

kept *0.0001* with fix-rate policy to enhance the learning capability of the model. The deep CNN model has been trained using the batch method where each batch consists of 8 images and the loss has been computed using the categorical cross entropy function.

Computational resources: This experiment has been conducted in Ubuntu 18.04.3 LTS server equipped with Intel Xeon Processor (2.39GHz) and Nvidia GRID V100D-16Q (16 GB each). The developed deep CNN model has been implemented by the Keras framework using the python programming language. Keras is one of the most powerful python-based deep learning libraries that run on top of tensorflow framework.

RESULTS AND DISCUSSION

The developed deep CNN model has been successfully trained, validated and tested on the collected dataset for identifying the MLB disease of maize. In this experiment, a total of 1547 images of maize leaves have been used. All the images are captured in real in-field conditions thus have complex backgrounds such as soil, other plant parts, human body parts and so on. The dataset contains images of leaves infected with maydis leaf blight disease as well as healthy leaves of maize. The dataset has been divided into three partitions such as training, validation, and testing for better measurement of the performance. The training data has been used for training the parameters of the model with features of the images, the validation data is used for the tuning of the trained parameters of the model and finally, the testing data is used for assessing the performance of the model.

In this experiment, two different train-validation-test configurations of the dataset have been used. The first configuration of the dataset consists of 70% training data, 15% validation data and 15% testing data. And the second configuration of the dataset is having 60% training data, 20% validation data and 20% testing data (in the rest of the paper 70:15:15 data and 60:20:20 data notations will be used to denote the two datasets). The model has been trained and validated for 1000 epochs (iterations) using both configurations of the dataset. The training and validation performance of the model on both the configurations of data has been shown in terms of accuracy and loss in the Fig 2 (A, B, C, D). The training accuracy and the training loss of both the data have shown similar behaviour, achieving optimum values after a few 100 epochs of the training iterations with very little fluctuation. Whereas, the validation accuracy and validation loss of both the data have many fluctuations during the training iterations. From data (Table 1) we can see that the 70:15:15 data achieved slightly higher validation accuracy of 99.57% than the 98.68% of the 60:20:20 data.

Table 1 Training and validation accuracies

Ratio	Training accuracy	Validation accuracy
60:20:20 data	99.97%	98.68%
70:15:15 data	99.98%	99.57%

Table 2 Testing performance of the model

Data	Classes	Precision	Recall	F1-score
60:20:20 data	Maydis leaf blight	0.9840	0.9892	0.9866
	Healthy	0.9836	0.9756	0.9796
	Accuracy		98.38%	
70:15:15 data	Maydis leaf blight	0.9930	0.9930	0.9930
	Healthy	0.9889	0.9889	0.9889
	Accuracy		99.14%	

The reason behind this difference is that the availability of a larger set of training samples to learn the features of the images in the 70:15:15 data than the 60:20:20 data. After successful training and validation, the model has been tested using the separate testing data for evaluating the performance of the trained model. The testing data wasn't used in the model training so that it can be used as new data for the model. There are 232 images in the 70:15:15 data and 309 images in the 60:20:20 data as testing data. In this study, the developed model has achieved the testing accuracy of 99.14% from the 70:15:15 data and 98.38% from the 60:20:20 data respectively. The class-wise precision, recall and f1 score for both the dataset have been given in Table 2. The experimental results (Table 2) suggest that model has been performed comparatively better in 70:15:15 data than that of 60:20:20 data.

In this experiment, a deep convolutional neural network model has been developed to identify the images of maydis leaf blight disease of maize crop. The architectural framework of popular 'GoogleNet' model has been used to develop this deep CNN model. All the images used in this experiment have been captured in real in-field condition from the standing crops from standing crop in different agricultural farms. We have trained the model with two train-validation-test configured dataset (70:15:15 data and 60:20:20 data). The 70:15:15 data has shown significantly better performance than the 60:20:20 data during validation as well as testing phase. The experiment showed that deep learning models give better results with larger training datasets. It is also possible to boost the identification capability of the model by introducing more images of the disease, altering the parameters of the model. This trained model can be saved and deployed in the sever. Then it can be integrated with mobile-based applications for serving the actual farm level. In the future, more number of diseases of maize crop will be explored for automated identification using the deep learning model and will be combined with this work.

ACKNOWLEDGEMENTS

The first author acknowledges the fellowship received from UGC-Maulana Azad National Fellowship (MANF) to undertake this research work as Ph D program. This work has been supported by National Agricultural Science Fund (NASF), ICAR, India.

REFERENCES

Boulent J, Foucher S, Théau J and St-Charles P L. 2019. Convolutional Neural Networks for the automatic identification of plant diseases. *Frontiers in plant science* **10**: 941.

DeChant C, Wiesner-Hanks T, Chen S, Stewart E L, Yosinski J, Gore M A, Nelson R J and Lipson H. 2017. Automated identification of northern leaf blight-infected maize plants from field imagery using deep learning. *Phytopathology* **107**(11): 1426–32.

Kamilaris A and Prenafeta-Boldú F X. 2018. Deep learning in agriculture: A survey. *Computers and electronics in agriculture* **147**: 70–90.

Krizhevsky A, Sutskever I and Hinton G E. 2012. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems* **1**: 1097–1105.

LeCun Y, Bengio Y and Hinton G. 2015. Deep learning. *Nature* **521**(7553): 436–44.

LeCun Y, Bottou L, Bengio Y and Haffner P. 1998. Gradient-based learning applied to document recognition. *(In) Proceedings of the IEEE*, 86th edn. Vol 11, November, pp 2278–2324.

LeCun Y, Kavukcuoglu K and Farabet C. 2010. Convolutional networks and applications in vision. *(In) Proceedings of 2010 IEEE International Symposium on Circuits and Systems,* May 30, pp. 253–56.

Malik V K, Singh M, Hooda K S, Yadav N K and Chauhan P K. 2018. Efficacy of newer molecules, bioagents and botanicals against maydis leaf blight and banded leaf and sheath blight of maize. *Plant Pathology Journal* **34**(2): 121–25.

Marwaha S, Haque M A, Deb C K, Arora A, Kumar M and Hooda K S. 2019. Maize disease classification using deep CNN model. *(In) Proceeding of 8th International Conference on Agricultural Statistics*, New Delhi, November 18-21.

Misra T, Arora A, Marwaha S, Chinnusamy V, Rao A R, Jain R, Sahoo R N, Ray M, Kumar S, Raju D and Jha R R. 2020. SpikeSegNet-a deep learning approach utilizing encoder-decoder network with hourglass for spike segmentation and counting in wheat plant from visual imaging. *Plant Methods* **16**(1): 1–20.

Mohanty S P, Hughes D and Salathe M. 2016. Using deep learning for image-based plant disease detection. *Frontiers in Plant Science* **7**: 1419.

Nigam S and Jain R. 2020. Plant disease identification using Deep Learning: A review. *Indian Journal of Agricultural Sciences* **90**(2): 249–57.

Priyadharshini R A, Arivazhagan S, Arun M and Mirnalini A. 2019. Maize leaf disease classification using deep convolutional neural networks. *Neural Computing and Applications* **31**(12): 8887–95.

Singh R and Srivastava R P. 2016. Southern corn leaf blight–an

important disease of maize: an extension fact sheet. *Indian Research Journal of Extension Education* **12**(2): 324–27.

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

Srivastava N, Hinton G, Krizhevsky A, Sutskever I and Salakhutdinov R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* **15**(1): 1929–58.

Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V and Rabinovich A. 2015. Going deeper with convolutions. (In) *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp 1–9.

Szegedy C, Vanhoucke V, Ioffe S, Shlens J and Wojna Z. 2016. Rethinking the inception architecture for computer vision. *(In) Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–26.

Zhang X, Qiao Y, Meng F, Fan C and Zhang M. 2018. Identification of maize leaf diseases using improved deep Convolutional Neural Networks. *IEEE Access* **6**: 30370–77.