Plant Disease Classification using Residual Networks with MATLAB

N.Alivelu Manga, P. Sathish

Department of ECE, ChaitanyaBharathi Institute of Technology,Hyderabad, Telangana, India

Article History: Received: 10 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online:28April 2021

Abstract

In a developing country like India, agricultural production mainly depends on smallholder farmers and more than 50% yield loss due to pests and various diseases affected by plants. Diseases can be managed by identifying the diseases as soon as it appears on the plant. In addition, the digital era makes the diagnosis information of any disease available at the fingertips. In other words, smartphones with high resolution cameras can aid in identification of a plant disease through images and thereby help in early diagnosis of the disease. This paper makes an attempt at resolving the present problem of undiagnosed plant diseases using Deep Learning for detecting and classifying the plant diseases using images of the plant. This helps the farmers to take necessary action to avoid the disease from aggravating without having to wait for an agriculturist to identify and resolve the problem.

In a nutshell, the paper aims at designing a Deep Learning model for the classification of an image of plant disease. The algorithm makes use of a Convolutional Neural Network with Residual Network architecture, commonly known as ResNet using Matrix Laboratory (MATLAB). The model will be trained using a public dataset of 54,305 colour images of diseased and healthy plants which when segregated result in 38 classes.

Keywords:Plant Disease, Images, Convolutional Neural Network, Residual Network, MATLAB.

Introduction

Research states that there is enough food being cultivated to feed the world yet, there are considerable people in various parts of the world who sleep with an empty stomach. According to recent reports, 14.8% of the Indian population is malnourished [1]. One of the most important factors that acts as a threat to food security is undiagnosed plant diseases. In addition to affecting food security they also pose a threat to the livelihood of small household farmers. It has been noted that when these plant diseases are identified at an early stage, the damage caused by the disease can be stopped from aggravating. In the recent past, with the widespread availability of access to the Internet, efforts to manage plant diseases have been supported by making information regarding diagnosis at each stage of the disease, available online. With rapidly changing technology, high resolution cameras have far-reaching effects in identification of a plant disease through images [2].

With a considerable understanding of the layers in a convolutional neural network and its architectures, it would be ideal to discuss the methodology followed to build the network and get the intended results. The steps essential in any machine learning or deep learning algorithm are preparing the database, deciding on the architecture, training and hyperparameter tuning to obtain the best possible set of metrics. These steps have been discussed in detail in the subsequent sections.

Today, AI has become an integral part of people's lives. With the help of these Deep Learning techniques images can be classified with minimal or no pre-processing required because of which Deep Learning has an edge over machine learning. Classification can be of two types, supervised and unsupervised. The paper aims at classifying images using Supervised Learning [3]. The process of classification involves two steps, namely, training and testing. Training process involves validation as well where the trained neural network is tested with a set of unseen images. Validation step helps avoid overfitting and sampling imbalance. The training process extracts characteristics of images belonging to a class and associates them as properties characteristic to that class. This process of extraction of properties is followed for each class depending on whether it is a binary-class classification or multiclass classification[4]. In the validation step, the trained network is used to classify images it had not been trained on. If considerably high, performance metrics such as accuracy, precision and recall are obtained, it means that the network can deployed for usage. Else, the training process must be re-initiated by passing different values to parameters such as mini-batch size, number of epochs, type of optimizer, learning rate and so on. The cycle of training and validation is followed until the desired accuracy, recall and precision are obtained[5]. This process of changing parameters to obtain better performance metrics is known as hyper-parameter tuning. Finally, the trained network is tested on the test dataset.

There are quite a few architectures in Convolutional Neural Networks that are very different and upon closer inspection it has been observed that Residual Neural Networks seem to make training easier and accurate using identity propagation [6].

The paper is organized into sub-topics that give an insight into how the database was prepared, trained and

tested. The subsequent sections discuss the results and the future enhancements that could be done to the algorithm.

Research Methodology

Preparing the Database

54,305 images of plant leaves were analyzed, which had a spread of 38 class labels assigned to them. Each class label is a crop-disease pair or a healthy crop, and an attempt was made to predict the crop-disease pair given just the image of the plant leaf. The 38 categories of various leaves such as Blueberry, Soybean, Apple, Peach, Orange and Tomato by considering the different diseases effect at various stages.Fig. 1 shows an example of a few crop-disease pairs from the dataset. The images were resized to 256 x 256 pixels to perform training, validation and testing on these images. In addition, the size of the kernels was chosen based on a set of masks generated by analysis of the color, lightness and saturation components ofdifferent parts of the images such that the size of the feature map remains same at each layer.

Fig. 1. Plant Disease Database

Architecture

A ResNet with 6 units of net-width 32 each and layers namely, 2D-Convolutional, Batch Normalization Layer for carrying out zero-centered normalization followed by Rectified Linear Unit activation was designed. These layers are used iteratively to design a Residual Network. The convolutional layer acts as a feature extractor which makes it ideal for image processing. Since the feature map obtained after each transformation will be different, batch normalization layer is used to normalize the output after each transformation hence, resulting in uniformity. It also reduces the chances of overfitting. The ReLU activation layer is used to induce non-linearities into the network.It is observed from the analysis result that while all the other layers are 2D matrices with multiple channels, the Fully Connected layer is a flattened layer with the number of neurons equal to the total number of classes. In this case it is 38. Fig. 2 shows the layer graph of the Residual Neural Network designed and Fig. 3 to Fig. 5 show parameters, like size, stride and padding of kernel used at each layer.

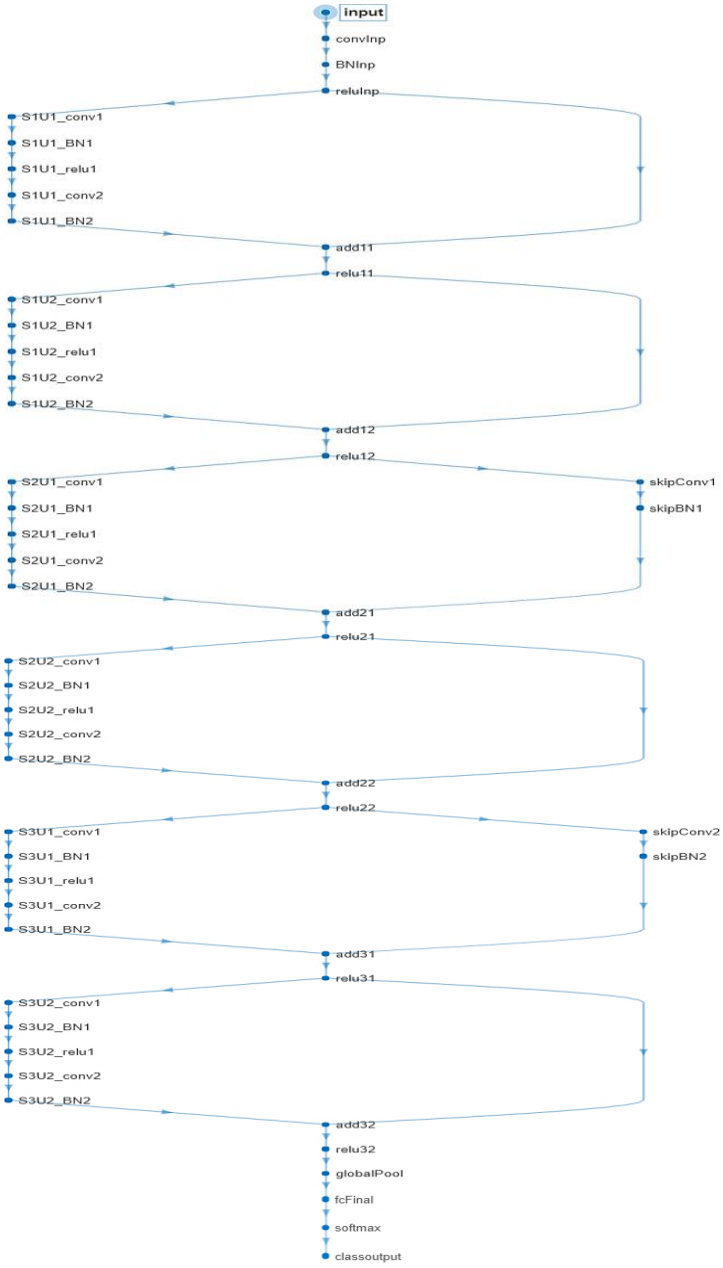

Fig. 2. Layer Graph

ANALYSIS RESULT

	NAME	TYPE	ACTIVATIONS	LEARNABLES
1	input 256x256x3 images with 'zerocenter' normalization	Image Input	256x256x3	-
2	convInp 32 3x3x3 convolutions with stride [1 1] and padding 'same'	Convolution	256x256x32	Weights 3x3x3x32 Bias 1x1x32
3	BNInp Batch normalization with 32 channels	Batch Normalization	256x256x32	Offset 1x1x32 Scale 1x1x32
4	reluInp ReLU	ReLU	256x256x32	-
5	S1U1_conv1 32 3x3x32 convolutions with stride [1 1] and padding 'same'	Convolution	256x256x32	Weights 3x3x32x32 Bias 1x1x32
6	S1U1_BN1 Batch normalization with 32 channels	Batch Normalization	256x256x32	Offset 1x1x32 Scale 1x1x32
7	S1U1_relu1 ReLU	ReLU	256x256x32	-
8	S1U1_conv2 32 3x3x32 convolutions with stride [1 1] and padding 'same'	Convolution	256x256x32	Weights 3x3x32x32 Bias 1x1x32
9	S1U1_BN2 Batch normalization with 32 channels	Batch Normalization	256x256x32	Offset 1x1x32 Scale 1x1x32
10	add11 Element-wise addition of 2 inputs	Addition	256x256x32	-
11	relu11 ReLU	ReLU	256x256x32	-
12	S1U2_conv1 32 3x3x32 convolutions with stride [1 1] and padding 'same'	Convolution	256x256x32	Weights 3x3x32x32 Bias 1x1x32
13	S1U2_BN1 Batch normalization with 32 channels	Batch Normalization	256x256x32	Offset 1x1x32 Scale 1x1x32
14	S1U2_relu1 ReLU	ReLU	256x256x32	-
15	S1U2_conv2 32 3x3x32 convolutions with stride [1 1] and padding 'same'	Convolution	256x256x32	Weights 3x3x32x32 Bias 1x1x32
16	S1U2_BN2 Batch normalization with 32 channels	Batch Normalization	256x256x32	Offset 1x1x32 Scale 1x1x32
17	add12 Element-wise addition of 2 inputs	Addition	256x256x32	-
18	relu12 ReLU	ReLU	256x256x32	-
19	S2U1_conv1 64 3x3x32 convolutions with stride [2 2] and padding 'same'	Convolution	128x128x64	Weights 3x3x32x64 Bias 1x1x64

Fig. 2. Analysis of Network Layers 1 to 19

It can be observed that the Convolution, Batch Normalization and Rectified Linear Unit Activation layer are used iteratively to form a deep neural network. Greater the depth, greater will be the accuracy of the network. Inorder to preserve the validation accuracy, An addition Layer is added. The Batch Normalization Layer is followed by ReLU activation layer and Addition layer alternatively. Presence of the addition layer makes the network a Residual Net. In the subsequent figures, the analysis of network layers, 20 to 36 and 36 to 54 can be observed.

ANALYSIS RESULT

	NAME	TYPE	ACTIVATIONS	LEARNABLES
20	S2U1_BN1 Batch normalization with 64 channels	Batch Normalization	128×128×64	Offset 1×1×64 Scale 1×1×64
21	S2U1_relu1 ReLU	ReLU	128×128×64	-
22	S2U1_conv2 64 3x3x64 convolutions with stride [1 1] and padding 'same'	Convolution	128×128×64	Weights 3×3×64×64 Bias 1×1×64
23	S2U1_BN2 Batch normalization with 64 channels	Batch Normalization	128×128×64	Offset 1×1×64 Scale 1×1×64
24	skipConv1 64 1x1x32 convolutions with stride [2 2] and padding [0 0 0 0]	Convolution	128×128×64	Weights 1×1×32×64 Bias 1×1×64
25	skipBN1 Batch normalization with 64 channels	Batch Normalization	128×128×64	Offset 1×1×64 Scale 1×1×64
26	add21 Element-wise addition of 2 inputs	Addition	128×128×64	-
27	relu21 ReLU	ReLU	128×128×64	-
28	S2U2_conv1 64 3x3x64 convolutions with stride [1 1] and padding 'same'	Convolution	128×128×64	Weights 3×3×64×64 Bias 1×1×64
29	S2U2_BN1 Batch normalization with 64 channels	Batch Normalization	128×128×64	Offset 1×1×64 Scale 1×1×64
30	S2U2_relu1 ReLU	ReLU	128×128×64	-
31	S2U2_conv2 64 3x3x64 convolutions with stride [1 1] and padding 'same'	Convolution	128×128×64	Weights 3×3×64×64 Bias 1×1×64
32	S2U2_BN2 Batch normalization with 64 channels	Batch Normalization	128×128×64	Offset 1×1×64 Scale 1×1×64
33	add22 Element-wise addition of 2 inputs	Addition	128×128×64	-
34	relu22 ReLU	ReLU	128×128×64	-
35	S3U1_conv1 128 3x3x64 convolutions with stride [2 2] and padding 'same'	Convolution	64×64×128	Weights 3×3×64×128 Bias 1×1×128

Fig. 4: Analysis of Network Layers 20 to 35

ANALYSIS RESULT

	NAME	TYPE	ACTIVATIONS	LEARNABLES
36	S3U1_BN1 Batch normalization with 128 channels	Batch Normalization	64×64×128	Offset 1×1×128 Scale 1×1×128
37	S3U1_relu1 ReLU	ReLU	64×64×128	-
38	S3U1_conv2 128 3x3x128 convolutions with stride [1 1] and padding 'same'	Convolution	64×64×128	Weights 3×3×128×128 Bias 1×1×128
39	S3U1_BN2 Batch normalization with 128 channels	Batch Normalization	64×64×128	Offset 1×1×128 Scale 1×1×128
40	skipConv2 128 1x1x64 convolutions with stride [2 2] and padding [0 0 0 0]	Convolution	64×64×128	Weights 1×1×64×128 Bias 1×1×128
41	skipBN2 Batch normalization with 128 channels	Batch Normalization	64×64×128	Offset 1×1×128 Scale 1×1×128
42	add31 Element-wise addition of 2 inputs	Addition	64×64×128	-
43	relu31 ReLU	ReLU	64×64×128	-
44	S3U2_conv1 128 3x3x128 convolutions with stride [1 1] and padding 'same'	Convolution	64×64×128	Weights 3×3×128×128 Bias 1×1×128
45	S3U2_BN1 Batch normalization with 128 channels	Batch Normalization	64×64×128	Offset 1×1×128 Scale 1×1×128
46	S3U2_relu1 ReLU	ReLU	64×64×128	-
47	S3U2_conv2 128 3x3x128 convolutions with stride [1 1] and padding 'same'	Convolution	64×64×128	Weights 3×3×128×128 Bias 1×1×128
48	S3U2_BN2 Batch normalization with 128 channels	Batch Normalization	64×64×128	Offset 1×1×128 Scale 1×1×128
49	add32 Element-wise addition of 2 inputs	Addition	64×64×128	-
50	relu32 ReLU	ReLU	64×64×128	-
51	globalPool 8x8 average pooling with stride [1 1] and padding [0 0 0 0]	Average Pooling	57×57×128	-
52	fcFinal 38 fully connected layer	Fully Connected	1×1×38	Weights 38×415872 Bias 38×1
53	softmax softmax	Softmax	1×1×38	-
54	classoutput crossentropyex with 'Apple___Apple_scab' and 37 other classes	Classification Output	-	-

Fig. 5: Analysis of Network Layers 36 to 54

Training

The database was split into three sets in the proportion, 70%, 15% and 15% for training, validation and testing respectively. Validation data was used to avoid overfitting and sub-sampling imbalance. The network was trained in mini batches of size 64 using Stochastic Gradient Momentum Optimizer. The network was initially trained for 87 Epochs with 593 iterations each and the checkpoint of each Epoch was saved so that the network in order to resume training from the desired point whenever required. Verbose logs consisting of the Epoch number, Iteration number, timestamp, Training Accuracy, Validation Accuracy, Training and Validation Loss were recorded after every 500 iterations. The network learned using Drop-out method and for better accuracy, the Learning rate was reduced by a factor of about 0.1 after 60 epochs. All the aforementioned were the initial parameters also known as hyperparameters. Using the training progress graph, and the verbose logs, the optimal number of epochs to avoid overfitting was obtained. Figure 3.6 shows the training progress graph. The first graph is a plot of training and validation accuracy against the number of epochs. The black dotted lines indicate the validation accuracy while the Blue line indicates the smoothed training accuracy.

The training and validation loss against the number of epochs is plotted in the second graph. The optimal number of epochs is obtained by identifying the point in the first graph where training accuracy goes on increasing while validations accuracy starts to decrease. In other words, it is the point where overfitting begins. In the graph, that point is the 62nd Epoch. Therefore, the model was re-trained for 62 epochs and the metrics were recorded. This way hyperparameter tuning was done to obtain the best possible set of metrics for the network.

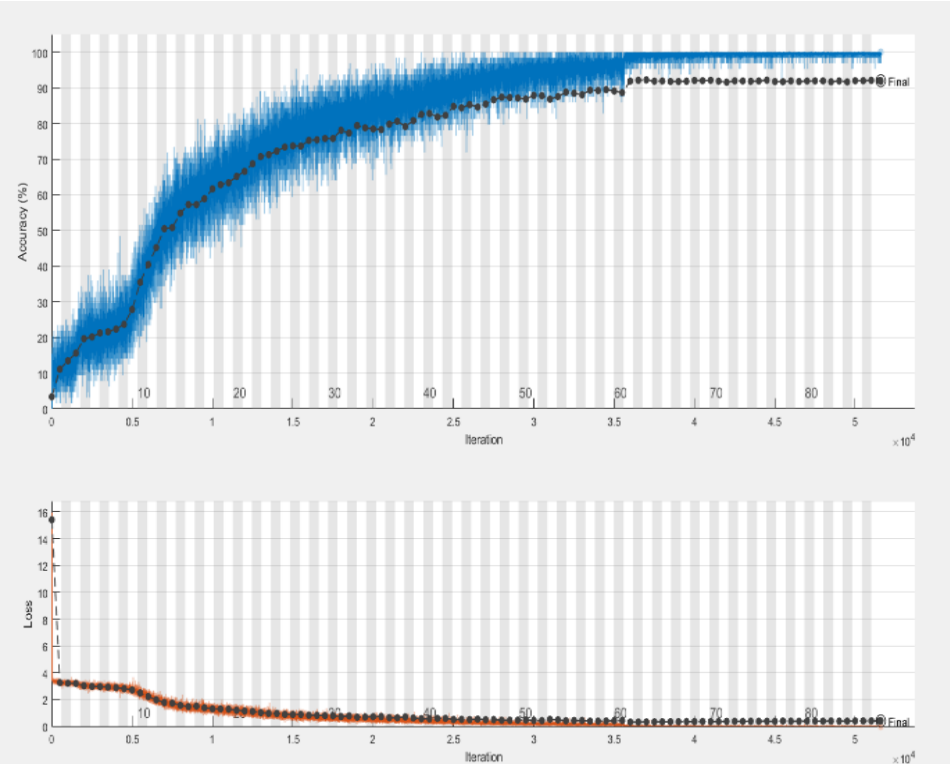

Fig. 6: Training Progress

Performance

The training accuracy for after re-training was 98.44% and the validation accuracy was 92.12%. According to experts, a 4% difference in training and validation accuracy is acceptable. Once all the steps in building a network had been completed and satisfactory levels of training and validation accuracy was achieved, the model was all set to be tested against unseen data, that is, the test set.

Results

The test set consisting of 8145 images were passed into the trained network without labels. The labels predicted for the images by the trained network were compared against the actual labels to get the accuracy, precision and recall. A confusion matrix was plotted for the same. The images that had been classified correctly were displayed with their crop-disease pair labels in green, while the images that had been wrongly classified was displayed in red. from the test set. Figure shows classified images picked at random.

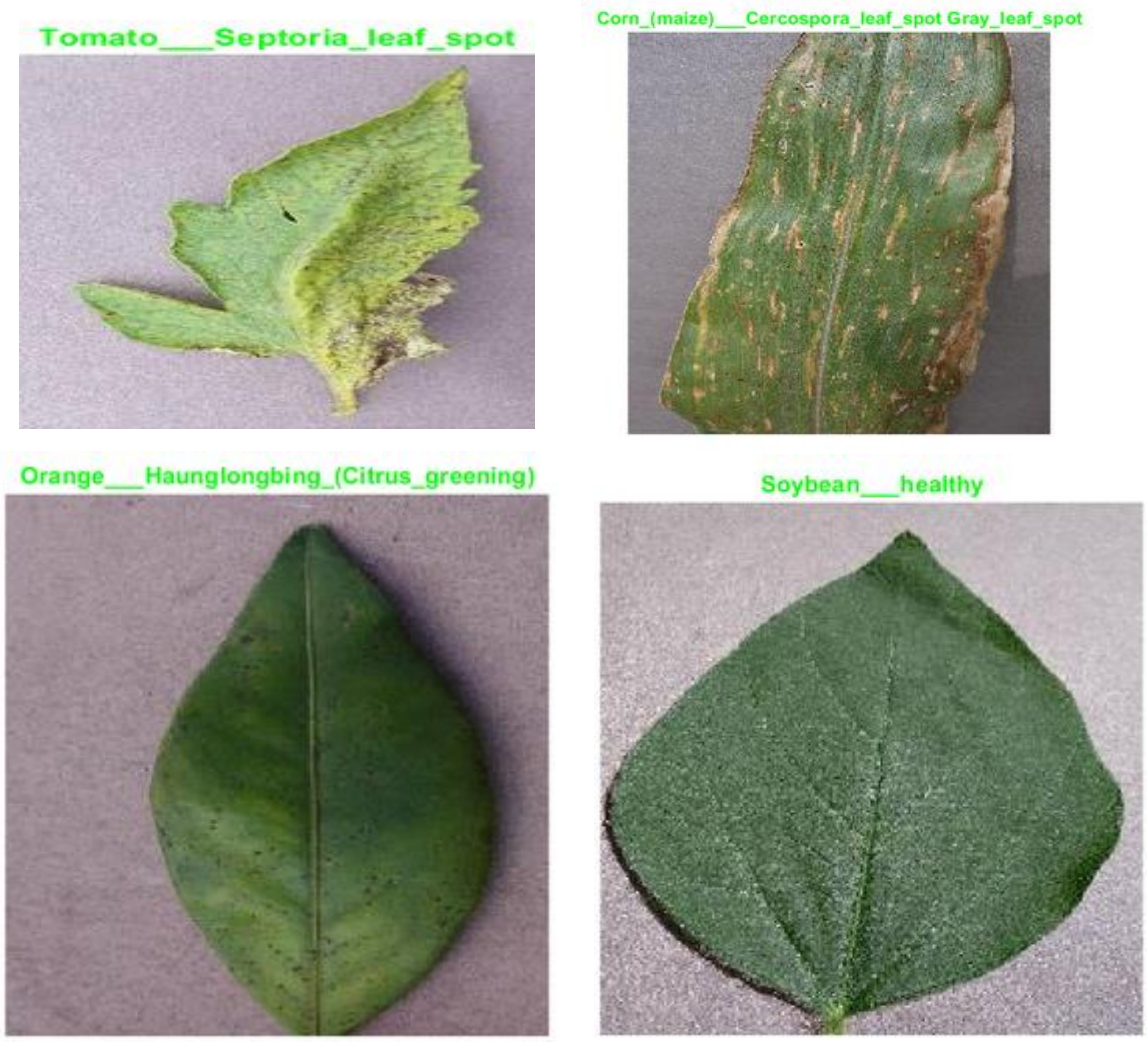

Fig. 7: Correctly classified crop-disease pairs

A correct classification indicates that the predicted label was same as the actual label. The labels were assigned based on probabilities. The output neurons in the fully connected layer assign probabilities to each class. The class with the highest probability after softmax layer is assigned to the image. The probability of top-5 classes predicted for one of the images in Figure 4.1 are shown using a bar-plot. Among all the classes Tomato_Septoria_leaf_spot had achieved the highest probability of 0.87 and hence the image has been classified as Tomato_Septoria_leaf_spot.

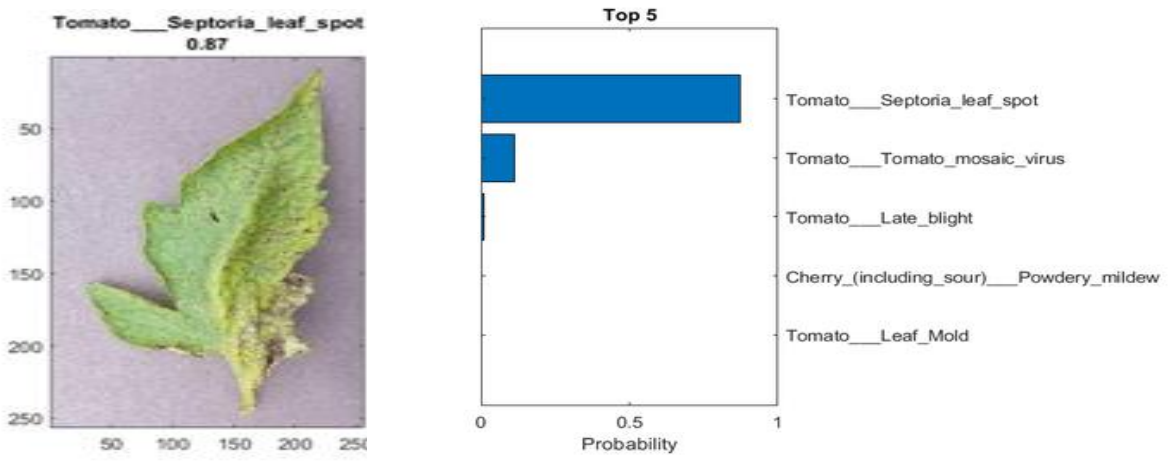

Fig. 8: Probability of top 5 classes

In order to get metrics such as accuracy, precision and recall a confusion matrix was plotted. It is a 38x38 matrix

with true or actual classes represented as rows and predicted classes represented as columns.

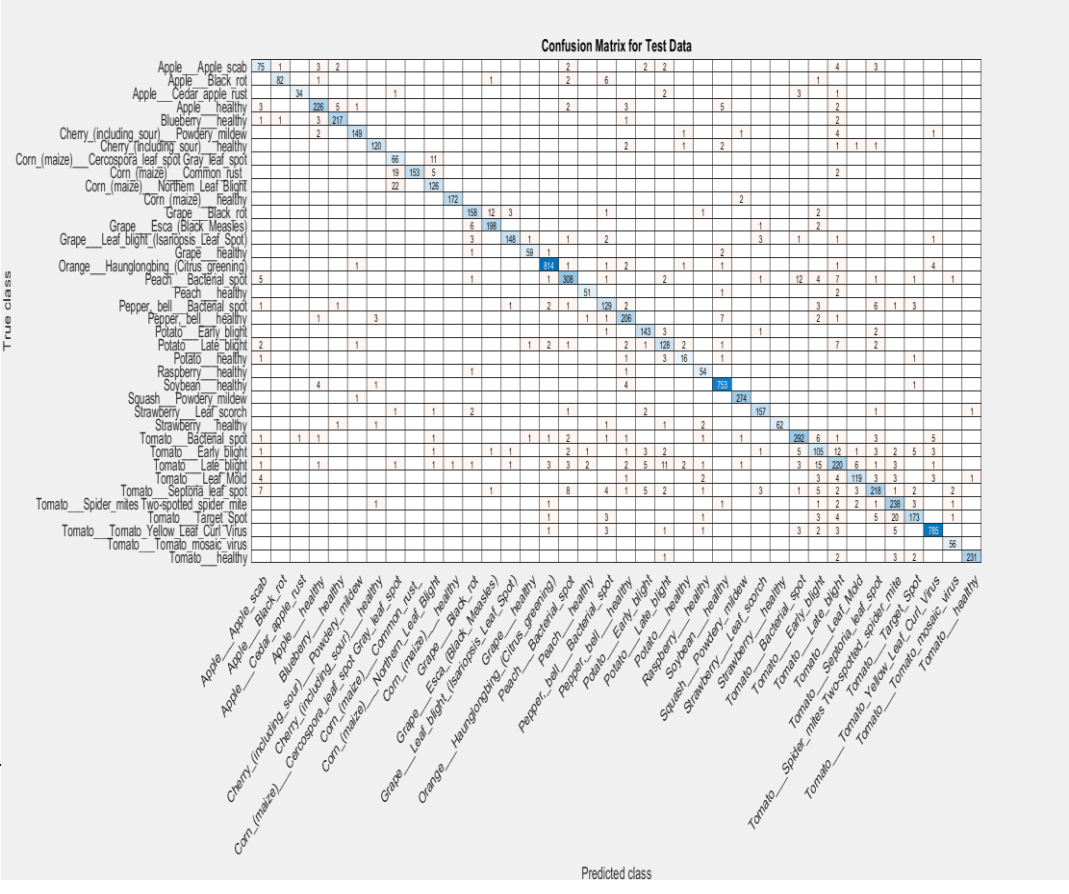

Fig. 9: Confusion Matrix

The diagonal elements in blue indicate the number of correctly predicted labels. The sum of all the diagonal elements divided by the sum of all the elements gives the test accuracy of the network.

$$\text{Accuracy} = \frac{\text{Sum of diagonal elements of Confusion matrix}}{\text{Total number of predictions}} * 100 = \frac{7331}{8145} * 100 = 90.0006\%$$

Equation 1: Accuracy

Therefore, in this case it was found to be 90%. Accuracy can be considered as the most intuitive performance measure but only in the case where the datasets are symmetric that is, they have nearly the same number of false positives and false negatives, which is not always the case. Hence,
precision recall and F1-score. Recall is the ratio of correctly predicted positive observations to the all observations in actual class. Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. F1 is a harmonic mean of the two.
The precision, recall and F1 score for this network were hence calculated as follows-

$$\text{Recall} = \frac{100}{\text{Total number of classes}} \sum \frac{\text{Correctly Predicted Positive Observations of each class}}{\text{All the observations in each actual class}} = \frac{34.1924}{38} * 100 = 89.98\%$$

Equation 2: Recall

$$\text{Precision} = \frac{100}{\text{Total number of classes}} \sum \frac{\text{Correctly Predicted Positive Observations in each class}}{\text{Total predicted positive observations}} = \frac{34.3672}{38} * 100 = 90.44\%$$

Equation 3: Precision

$$\text{F1} - \text{Score} = \frac{2 * \text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}} = \frac{2 * 90.44 * 89.98}{90.44 + 89.98} = 90.21$$

Equation 4: F1-Score

Therefore, the precision and recall obtained for the network are 90.44% and 89.98%. The F1-score was found to be 90.21%.
Precision is important in scenarios where cost of a false positive that is wrongly predicted as positive observations is high. On the contrary, Recall is important when cost associated with a false negative is high. In the case of plant disease classification, the network has been developed for small household farmers hence, a plant disease gone unidentified will result in huge loss for the farmer. Hence, our interest in this project is recall and F1-score since a very low precision is also undesirable. Due to the hyperparameter tuning the precision and recall and thereby F1-score obtained are nearly the same and close to validation accuracy. Therefore, the trained network can be deployed as a RESTful API using POST method to help farmers across the country.

Conclusion

The challenging task in agricultureis crop protection it requiresan in-depth knowledge of the crops grown at various stages and effect of various diseases in each stage. In the paper specialized deep learning model has been designed, based on the architectures of the specific convolutional neural networks, for the detection of plant diseases through leaves images of healthy or diseased plants.The whole development of the algorithm was done using MATLAB tool. The algorithm deals with training of the dataset andits classification of various images. Thevarious test input images were compared with the trained dataset for the detection and prediction analysis. Here, Supervised Learning Technique was used for precise accuracy. The network was found to have an accuracy of 90%, precision, recall and F-1 score of 90.44%, 89.98% and 90.21% respectively. Precision, Recall and Accuracy are nearly equal and considerably high. Therefore, the designed network not only preserves accuracy but also deals with the problem of vanishing gradient. These performance metrics show that the model is optimal for deployment. As mentioned in the abstract, only images are being used for processing the plants without disturbing their environmental harmony, there is no environmental hazard issue with this proposed methodology of plant disease classification. Therefore, it is fully completely safe and also it helps in growing of the plants effectively with low cost and timely diagnosis of any disease.

Recommendations and Future Work

Further thiswork can be extended for other plant leaves in various regions. . For the deployment of the

end product the high performance processors will be recommended for the further fast processing.The end product would be accurately identifying the plant disease. Larger set of data would be provided for training network. OpenCV can be used for analysing images which is analogous to the Image Processing Toolbox in MATLAB. In addition, an attempt can be made at displaying the results of classification and diagnosis of the identified disease in the language known to the farmer. Therefore, the farmer has to just take an image of the leaf, and upload it to the server or Virtual Machine through an application or GUI where the back-end processing will do the required classification and give necessary measures better plant growth

References

[1] Huang, J., Rathod, V., Sun, C.; Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al. Speed/accuracy trade-offs for modern convolutional object detectors. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 22–25 July 2017.

[2] Jiang Lu, Jie Hu, Guannan Zhao, Fenghua Mei, Changshui Zhang, An in-field automatic wheat disease diagnosis system, Computers and Electronics in Agriculture 142 (2017)
369–379.

[3] Konstantinos P. Ferentinos, Deep learning models for plant disease detection and diagnosis Computers and Electronics in Agriculture 145 (2018) 311–318.

[4] Rumpf, T.; Mahlein, A.-K.; Steiner, U.; Oerke, E.-C.; Dehne, H.-W.; and Plumer, L. 2010. Early detection and ¨ classification of plant diseases with support vector machines based on hyperspectral reflectance. Computers and Electronics in Agriculture 74(1):91–99. [5] Sankaran, S.; Mishra, A.; Ehsani, R.; and Davis, C. 2010. A review of advanced techniques for detecting plant diseases. Computers and Electronics in Agriculture 72(1):1–13. Soderkvist, O. 2001. Computer vision classification of ¨ leaves from swedish trees.

[6] Srivastava, N.; Hinton, G. E.; 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):1929–1958.