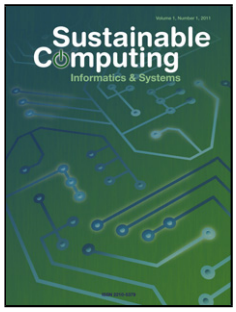

Development of Efficient CNN model for Tomato crop disease identification

Mohit Agarwal, Suneet Kr. Gupta, K.K. Biswas

PII: S2210-5379(20)30134-7

DOI: https://doi.org/10.1016/j.suscom.2020.100407

Reference: SUSCOM 100407

To appear in: *Sustainable Computing: Informatics and Systems*

Received Date: 27 November 2019

Revised Date: 8 May 2020

Accepted Date: 9 June 2020

Please cite this article as: Mohit Agarwal, Suneet Kr. Gupta, K.K. Biswas, Development of Efficient CNN model for Tomato crop disease identification, *<![CDATA[Sustainable Computing: Informatics and Systems]]>* (2020), doi: https://doi.org/

Development of Efficient CNN model for Tomato crop disease identification

Mohit Agarwal*, Suneet Kr. Gupta, K.K. Biswas

Deptt. of Computer Science Engineering, Bennett University, Greater Noida 201310, India

Abstract

Tomato is an important vegetable crop cultivated worldwide coming next only to potato. However, the crop can be damaged due to various diseases. It is important for the farmer to know the type of disease for timely treatment of the crop. It has been observed that leaves are clear indicator of specific diseases. A number of Machine Learning (ML) algorithms and Convolution Neural Network (CNN) models have been proposed in literature for identification of tomato crop diseases. CNN models are based on Deep Learning Neural Networks and differ inherently from traditional Machine Learning algorithms like k-NN, Decision-Trees etc. While pretrained CNN models perform fairly well, they tend to be computationally heavy due to large number of parameters involved. In this paper a simplified CNN model is proposed comprising of 8 hidden layers. Using the publicly available dataset PlantVillage, proposed light weight model performs better than the traditional machine learning approaches as well as pretrained models and achieves an accuracy of 98.4%. PlantVillage dataset comprises of 39 classes of different crops like apple, potato, corn, grapes etc. of which 10 classes are of tomato diseases. While traditional ML methods gives best accuracy of 94.9% with k-NN, best accuracy of 93.5% is obtained with VGG16 in pretrained models. To increase performance of proposed CNN, image pre-processing has been used by changing image brightness by a random value of a random width of image after image augmentation. The proposed model also performs extremely well on dataset other than PlantVillage with accuracy of 98.7%.

Keywords: Convolution Neural Network (CNN), Augmentation, Max Pooling, Machine Learning, Image pre-processing

1. Introduction

Identifying diseases from images of the plant leaf is one of the most important research areas in precision agriculture. With better computing power, image processing [1, 2] and

*Corresponding author: Mohit Agarwal

Email addresses: ma8573@bennett.edu.in (Mohit Agarwal), suneet.gupta@bennett.edu.in (Suneet Kr. Gupta), kanad.biswas@bennett.edu.in (K.K. Biswas)

the latest Neural Network research plant growth and protection practices can be greatly improved. There are many artificial intelligence (AI) approaches that are prevalent for detecting and classifying plant diseases. The most common AI approaches are Neural Networks, Logistic Regressions, Decision Trees, Support Vector Machines (SVM) [3, 4], k-Nearest Neighbors (k-NN), Naïve Bayes [5] and Deep Convolutional neural networks (Deep CNN) [6, 7].

Rumpf et al. [4] have demonstrated the use of SVMs for detecting 3 types of diseases in sugar beet root using leaves images. SVM has also been used by Mokhtar et al. [3] to classify two types of tomato plant diseases and obtained an accuracy of 92%. Johannes et al. [5] used the Naive Bayes technique for the identification of three diseases of wheat leaves.

Other Machine Learning techniques are evaluated in [8] by Yang and Guo who provide a review of machine learning techniques such as Naıve Bayes classifier, Support Vector Machine (SVM), K-Means clustering, Artificial Neural Networks (ANN), Decision Trees and Random Forests to identify plant diseases from leaf images. The authors use machine learning for the identification of genes involved in plant-pathogen (disease-causing bacteria or virus) interactions.

Recently several researchers have reported that Deep Learning is a better approach to achieve high accuracy in plant disease identification [6, 9, 10]. Generally, researchers have used transfer learning on pre-trained models of other domains and reported good results in disease classification.

The most common deep learning method for analyzing the image data is CNN (Convolution Neural Network). Mohanty et al. [6] have used popular CNN models named as AlexNet [11] and GoogLeNet [12] for prediction of classes of plants and diseases in images from PlantVillage dataset [1] reported in [13]. Ferentinos [10] has shown the use of pre-trained models Alexnet and VGG (Visual Geometry Group) for the classification of a plant as well as disease from 25 different plants and 58 distinct classes of crop and disease combined. Authors used the PlantVillage dataset augmented with other images and report an accuracy of 99.53% with VGG pre-trained model but very poor accuracy for images from a different database.

Transfer learning has been used in Zhang et al. [14] for identifying maize leaf disease from 9 types of maize leaves. Using max-max-ave pooling in 3 CNN hidden layers authors report an accuracy of 98.9% for GoogleLeNet and 98.8% for Cifar10 neural networks. Usage of transfer learning is also reported by Rangarajan et al. [9] who have employed AlexNet and VGG16 to train and classify tomato plant disease images from the PlantVillage dataset. They have shown that AlexNet gives around 97.29% and VGG16 gives around 97.49% accuracy in prediction on 373 test images in each class.

In [15] Fuentes et al. have proposed a Region-based CNN(R-CNN) and Region-based FCN(R-FCN) to identify the disease and bounding box of the diseased part in tomato leaves images. They have manually annotated the diseased portion of the leaves and also used augmentation to prevent overfitting for any particular class and used the pre-trained

[1]Dataset can be downloaded from: https://github.com/spMohanty/PlantVillage-Dataset

models VGG [16] and ResNet [17] for their CNN design. Best Mean accuracy of 83.06% was obtained for R-CNN and 85.98% for R-FCN.

Most authors working in plant disease identification have been using pre-trained models of other domains, and applying transfer learning for their specific tasks. Many authors have been trying to identify diseases of multiple crops simultaneously [6, 10]. However, it must be noted that pre-trained models were developed for situations where the number of classes is huge. For plant diseases, the number of classes is very small, and the use of such huge models is an overkill. Although some attempts have been made to develop low order CNN models for plant disease identification [18, 19, 20], the accuracy reported is not high. Wang et al. [18] have trained a shallow network with 2 to 10 convolutional layers on publicly available PlantVillage dataset. They used two fully connected layers and final softmax layer with 4 classes of Apple diseases and reported that the best accuracy of 79.3% was obtained for 8 convolution layers. They have reported that using transfer learning based on pre-trained models (VGG16, VGG19, Inception-V3, ResNet50) the testing accuracy varied from 87% to 93%. Khamparia et al. [19] have proposed a small model using two convolution layers, and tested on 10 disease classes of mixed crops and reported accuracy of 93.7% in 5 epochs. Hu et al. [20] have also developed a CNN model from scratch and reported accuracy of 92.5% for 4 tea leaf disease classes, using basic data augmentation.

In this paper, a lightweight CNN model has been proposed for the identification of 9 types of diseases in tomato plants. The proposed CNN model does not use a complex network structure of pre-trained models having a large number of hidden layers and parameters and has the advantage of reduced storage capacity and fast response while maintaining the same level of accuracy as reported in the literature. It is shown that this model fares very well compared to standard Machine Learning approaches, as well as popular pre-trained CNN models. To illustrate the robustness of the proposed model, it has been shown that while the model has been trained on the PlantVillage dataset, it performs exceedingly well on an altogether different dataset of tomato crops. In this paper, a novel method of augmentation has been introduced, which supplements the training dataset by creating new leaf images under varied lighting conditions. This aims to simulate actual field conditions where the brightness of leaves could vary depending on the position of the sun, or because of shadows cast by other leaves.

In this paper performance of the proposed CNN model is compared with those of the traditional Machine Learning (ML) techniques and pre-trained CNN models, and it has been shown that the model achieves comparable accuracy while needing much less storage space. The main contributions in this paper are a better performance by the proposed CNN model over pre-trained models, better performance over traditional Machine Learning methods, achieving state of art accuracy, and good performance on testing with different datasets.

The rest of paper is organized as follows: Brief introduction to traditional machine learning methods and deep convolutional neural networks is covered in section 2. In section 3 proposed CNN model is presented. The dataset used in the proposed work is discussed in section 4. The details of experimental results and analysis are presented in section 5. The main findings of this paper are discussed in section 6 followed by conclusions in section 7.

2. Methods Used

In this section, traditional Machine Learning algorithms have been briefly discussed which are used for plant disease(s) identification, followed by a brief overview of different types of Convolution Neural Networks.

2.1. Traditional Machine Learning Approaches:

These methods use a feature set which is in form of an input vector. The feature values are calculated using various Image Processing techniques for image classification problems.

2.1.1. Support Vector Machine (SVM):

This is used for Binary classification. Main idea governing SVM [1] is to construct a separating hyperplane given a set of training data such that separation between positive and negative samples is maximized.

Let x be input feature vector then equation of hyperplane can be written as described by Haykin [21]:

$$w^T x + b = 0 \tag{1}$$

Here: w is a vector variable of same size as input feature vector and b is the offset deciding the equation of hyperplane.

For samples with output = 1, the equation becomes:

$$w^T x_i + b > 0 \tag{2}$$

For samples with output = -1, the equation becomes

$$w^T x_i + b \leq 0 \tag{3}$$

When separating distance is tried to be maximized then on applying differential calculus to maximize the separating distance and minimize overall classification error in training data, following equations are obtained:

Maximize the objective function:

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j x_i^T x_j \tag{4}$$

subject to:

$$\sum_{i=1}^{N} \alpha_i d_i = 0 \tag{5}$$

and

$$\alpha_i \geq 0 \tag{6}$$

Where i = 1,2,.....N

Here:

α - Lagrangian multiplier; d_i - output value = -1 or 1; x - input feature vector

Finally w and b are obtained using the equation:

$$w_0 = \sum_{i=1}^{N} \alpha_i d_i x_i \tag{7}$$

$$b_0 = 1 - w_0^T x^{(s)} \tag{8}$$

Where $x^{(s)}$ is the positive side support vector

As clearly visible the equation is depending only on input data. Some variations of the above linear kernel are possible with a radial basis function. In radial basis function, non-separable input can be mapped to linearly separable points in hyperspace. This is one of the most commonly used machine learning techniques.

Other traditional Machine Learning techniques are as described below:

a) Naive Bayes: The technique [22] is based on estimating the probability of a class given a set of feature (feature vector x). The probability is computed by the formula:

$$P(y|x_1, x_2, ...x_n) = \frac{P(x_1|y)P(x_2|y).....P(x_n|y)}{P(x_1)P(x_2).....P(x_n)} \tag{9}$$

where y is the output class; and x_1, x_2,.....x_n are components of the feature vector.

Here probability of class y given a feature vector x is calculated using the product of probability of each feature given class y divided by the product of probabilities of each feature. The denominator being constant only the numerator is considered to compute the probabilities for each class and maximum probability for any class provides the output class for the input x.

b) k-NN: When the class of an input feature vector is decided based on the weighted distance of its distance from k nearest neighbors, it is called k-Nearest Neighbor algorithm [23]. The output y can be given as in equation below:

$$y = \sum_{i=1}^{k} W(x_0, x_i) y_i \tag{10}$$

where W is weight for the i^{th} neighbor and y_i is the output for i^{th} nearest neighbor.

Weight is calculated using the formula:

$$W(x, x_i) = \frac{exp(-D(x, x_i))}{\sum_{i=1}^{k} exp(-D(x, x_i))} \tag{11}$$

Here $D(x, x_i)$ can be Euclidean or Manhattan distance between points.

c) Decision Trees: In this model a tree is formed in form of flow chart [24] where each node is created based on information gain from each attribute, and its child links are the output of tests and leaves are class labels. The inputs are divided into subsets which are further divided till leaves, with the proper set having a correct class label, are reached. This process results in the same decision tree for a given set of labeled inputs. Decision Trees can be split while training using entropy, Gini Index, or reduction in variance. To calculate the entropy of a node following formula is utilized:

$$E(S) = \sum_{i=1}^{c} -p_i log_2(p_i) \tag{12}$$

Here p_i is the probability of an event. The event node needs to be split so that Information gain is maximum which depends on the decrease in entropy.

d) Logistic Regression: This model [25] is similar to a single layer neural network with coefficients of input vectors learned using gradient descent with cost function similar to the backpropagation cost function of Neural Networks given in equation (16). In this case the probability of output Y can be given using the formula:

$$P(Y) = \frac{1}{1 + e^{-(a+bX)}} \tag{13}$$

where the constants a and b are constants which are learned while training. When $P(Y) < 0.5$ then the output Y is 1 else it is 0. As seen this equation is similar to that of the sigmoid function.

2.1.2. Feature set used for ML algorithms:

The traditional Machine Learning methods used for comparison in this paper are SVM, Decision Tree, Logistic Regression, k-NN, and Naıve Bayes. Features such as Hu-moments, Haralick features, LBP features, and HSV features have been used to evaluate the performance of all traditional Machine Learning algorithms. These are described below. The features passed to traditional machine learning algorithms were:

- **Haralick Features:** These are based on the texture of the image and generated using Gray Level Co-occurrence Matrix (GLCM) using one of entropy as in equation (14), homogeneity or energy of these matrix element values.
- **Hu-features:** These are features of the object in the image and generated using centralized moments as in equation (15).
- **HSV-features:** These are made after conversion of RGB image to HSV and then calculating a histogram of HSV values which is converted to vector by flattening.
- **LBP-features:** LBP (Local Binary Pattern) is also a powerful texture-based feature calculated by comparing a pixel with 8 neighboring pixels.

$$E = -\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} p(i,j) log_b p(i,j) \tag{14}$$

In equation (14): E is the entropy; n, b are gray levels in image mostly 255 and $p(i,j)$ is the probability of two pixels separated by the specified offset having intensities i and j.

$$\mu_{pq} = \sum_{x}\sum_{y}(x-\bar{x})^p(y-\bar{y})^q f(x,y) \tag{15}$$

In equation (15): μ_{pq} are the centralized moments; x, y are pixel co-ordinates and $f(x,y)$ are pixel intensities at these co-ordinates. Here p = 0,1,2,3 and q = 0,1,2,3

2.2. Deep Convolution Neural Networks (CNN):

Deep CNN [6, 7, 26, 27] are specialized Feed forward Neural Networks which generally take images as input. Instead of explicitly extracting the image features to be fed into the network, the features are automatically extracted by convolution layers.

2.2.1. Forward Propagation in CNN:

The images which are given as input to CNN are resized to some fixed size and convolved over image processing filters. A Deep CNN model is shown in Fig. 1, which accepts the image of size 128 128 and applies 32 convolution filters on it. The role of the convolution filter is to extract the important features from the training dataset.

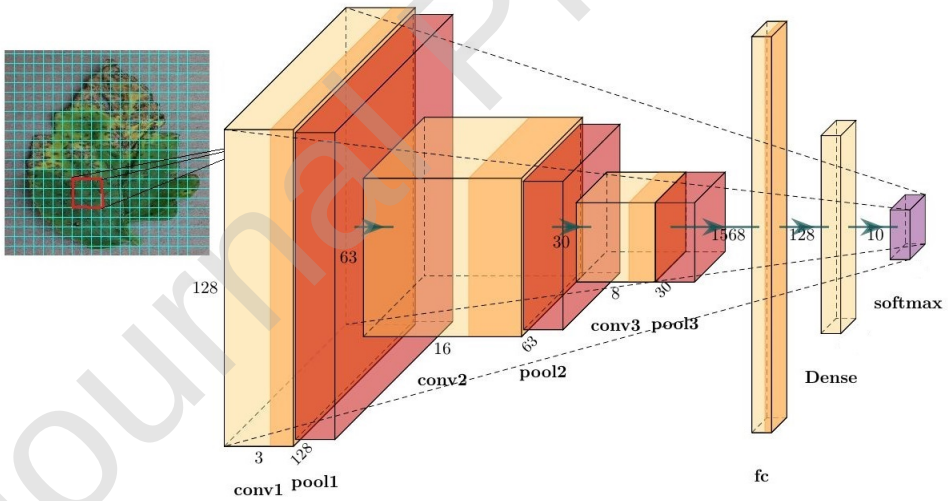

Figure 1: Proposed CNN model.

2.2.2. Back Propagation in CNN:

The Deep CNN model starts with random weights and during back propagation, the loss is calculated with the help of equation (16) as given below. As per the calculated loss, the weights and biases are changed with respect to the gradient. The whole process (forward and backward propagation) is iterated for a certain number of epochs to minimize the average loss.

$$L\left(a, y\right) = -\left(y \log(a) + (1-y)\, log\left(1-a\right)\right) \tag{16}$$

Here L is the loss, y is expected output and a is predicted output for a single test case.

2.2.3. Tuning of hyperparameters:

In Deep CNN there are several hyperparameters like a number of epochs, hidden layers, hidden nodes, activation function, drop out, learning rate, batch size, etc. which affect the performance of the model. In the hyperparameters tuning, experiments are repeated with a different number of hidden layers and epochs and different activation functions or learning rate. After fine-tuning of the model optimal state is quickly reached with the best accuracy.

2.3. Performance evaluation metrics:

The performance of the proposed model was compared with other traditional machine learning methods and pretrained CNN models based on following evaluation metrics as explained by Sokolova and Lapalme [28] and also by Tharwat [29]:

- **Accuracy:** It is the main criterion for evaluating the efficiency of a model. A better model will have higher accuracy than the other model. It is calculated using equation (17).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{17}$$

where TP - True Positive, TN - True Negative, FP - False Positive and FN - False Negative

- **F1-score:** It is also a popular method of performance comparison of different learning methods and is the harmonic mean of precision and recall. F1-score is calculated using equation (18).

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{18}$$

Here F_1 is F1-score. The formula for precision and recall are as given in equation (19) and (20).

$$Precision = \frac{TP}{TP + FP} \tag{19}$$

$$Recall = \frac{TP}{TP + FN} \tag{20}$$

Here TP - True Positive, TN - True Negative, FP - False Positive and FN - False Negative

- **Area Under the Receiver Operating Characteristic (AUC – ROC) curve:** It is a plot between TPR (y-axis) and FPR (x-axis) (refer equation (21) and (22)), and serves a measure to show how good a given machine learning method can distinguish between positive and negative classes under different threshold settings. A good model would have the AUC (Area Under the Curve) close to 1, while a value near about 0.5 indicates that the model is not suitable for classification. A good model will have high y-axis values on the ROC curve to show higher true positives.

$$TPR = \frac{TP}{TP + FN} \tag{21}$$

$$FPR = \frac{FP}{FP + TN} \tag{22}$$

Here TPR – True Positive Rate and FPR = False Positive Rate

2.4. Pre-trained CNN models

The pre-trained models are already designed CNN models and trained for specific datasets. These are generally used using transfer learning for other image classification problems. Three such models are described below:

2.4.1. VGG16

VGG 16 [16] is the CNN architecture created by VGG (Visual Geometry Group, University of Oxford). In brief its architecture has an input image size of 224 × 224. 3 × filters are used, but the padding is done to keep the resolution of intermediate outputs the same. It has 13 convolution layers and 3 Dense layers. Activation function used is ReLU in all layers. 2 penultimate layers have 4096 hidden nodes each and the final layer has 1000 output nodes equal to the number of classes in the ILSVRC (ImageNet Large Scale Visual Recognition Competition).

2.4.2. InceptionV3

Inception V3 [30] is a 42-layer deep learning network with fewer parameters. The reduction in parameters is done with help of factorizing convolutions. For example, a 5 × 5 filter convolution can be done by two 3 × 3 filter convolutions. The parameters in this process reduces from 5 × 5 = 25 to 3 × 3+3 × 3 = 18. Thus, it brings 28% reduction in the number of parameters. With a smaller number of parameters, the model will less overfit and thus increase the accuracy.

2.4.3. Mobilenet

Mobilenet [31] architecture uses depth-wise separable convolutions for making lightweight deep convolution neural networks. Two global hyperparameters are introduced: width multiplier and resolution multiplier which efficiently tradeoffs between accuracy and storage space. It is designed for usage on mobile devices with less storage needed. Mobilenet has 27 convolution layers followed by an average pooling layer and then a fully connected layer to transform the 2-D signal to 1-D and a final softmax output layer.

3. Proposed CNN model

Although a large number of plant disease researchers have used pre-trained models, it must be realized that these models were developed with a large number of layers to handle situations where the number of classes is very large. More layers imply a higher degree of the polynomial being used in the model. These, in turn, need very large storage to handle the parameters, and also very large computational time. It may be noted that for most plants the number of disease classes is less than 15. The use of pre-trained models on such plant data is likely to lead to overfitting and to produce erroneous results. On the other hand, a simple linear model with no hidden layer would lead to underfitting and would also produce erroneous results. It is therefore suggested that a lighter version of the CNN model be developed which would be more appropriate for a small number of classes.

The proposed light weight CNN model was obtained by compressing VGG16 by the following process:

- Get the weights of filters from hidden convolution layers and make the sum of its absolute values and sort them and pick top most relevant weights and make a list with 1 in the position where weights are substantial and 0 where weights are negligible.
- Count the number of ones in the list created in the above step and make a new CNN model with those number of nodes/filters and copy the weights from previous CNN where there was 1 in the list of that hidden layer.

After completing this process of new CNN creation and copying weights train the network further for 100-500 epochs and check it's accuracy will be more than previous CNN accuracy. The weight file will also be lighter due to lesser hidden nodes. Similarly lesser hidden layers can be kept (example in VGG16 there are consecutive convolution layers which can be lessened to single convolution layer). In this way, a new CNN gets created where weights can be copied from original CNN and trained further and tested. In this process, 94 MB weight file of VGG16 was reduced to 9 MB in the new model and convolution layers were reduced from 13 to 5, and accuracy increased from 93.5% to 94% when trained by transfer learning for 1000 epochs. In next step model was tested with variants of this compressed model and the optimum proposed CNN architecture was reached with the best accuracy.

These variants of proposed optimum CNN architecture had 2 - 4 convolution layers and 2 hidden dense layers. Experiments were conducted to arrive at the best low order convolution model. These are described in section 5.1.2. The input convolution layer consists of 32

kernels of size 3 × 3. The input leaf images are resized to 128 × 128 and 3 RGB channels are separately passed to the input convolution layer. For the most optimum 3 layer model, 2 more convolution layers are included with kernels of sizes 16 and 8 respectively. After each convolution layer, a max-pooling layer is also present to reduce the dimension of the image to half its size. After 3 convolution layers, a flattening layer is added to convert 2-D input to 1-D. Following it is a Dense hidden layer of 128 neurons. Finally, a softmax layer is included which outputs 10 possible classes. The models are implemented on the NVIDIA DGX v100 machine. The machine is equipped with 40600 CUDA cores, 5120 tensor cores, 128 GB RAM, and 1000 TFLOPS speed. The optimized CNN architecture is given in Table 1. A diagrammatic view of the proposed model is provided in Fig. 1.

Table 1: Model summary of proposed CNN architecture.

Layer (type)	**Input size**	**Output size**
Conv2d	128 × 128 × 3	126 × 126 × 32
Maxpool	126 × 126 × 32	63 × 63 × 32
Conv2d	63 × 63 × 32	61 × 61 × 16
Maxpool	61 × 61 × 16	30 × 30 × 16
Conv2d	30 × 30 × 16	28 × 28 × 8
Maxpool	28 × 28 × 8	14 × 14 × 8
Flatten	14 × 14 × 8	1568
Dense	1568	128
Dropout	128	128
Dense	128	10

The hyper parameters used in proposed model are given in Table 2.

Table 2: Hyper parameters for proposed CNN model.

Hyperparameter	Description
No. of convolution layer	3
No. of max pooling layer	3
Drop out rate	.5
Network weight initialization	Glorot uniform
Activation function	Relu
Learning rate	0.001
Momentum	0.999
Number of epoch	5000
Batch size	64

4. Dataset

The proposed model has been implemented on the PlantVillage dataset [13]. The dataset includes around 55,000 leaf images of 14 crops, such as tomatoes, potatoes, grapes, etc. Healthy and diseased crops constitute 39 classes of data. As far as the tomato crop is concerned, the dataset consists of 9 folders of tomato disease classes and 1 folder of healthy leaf images. However, the number of samples in different folders is not uniformly distributed. They range from 200 to 4000 samples. The dataset provides 14,529 labeled training images and 3,631 labeled validation images for tomato crops. The sample images of tomato crops are depicted in Fig. 2.

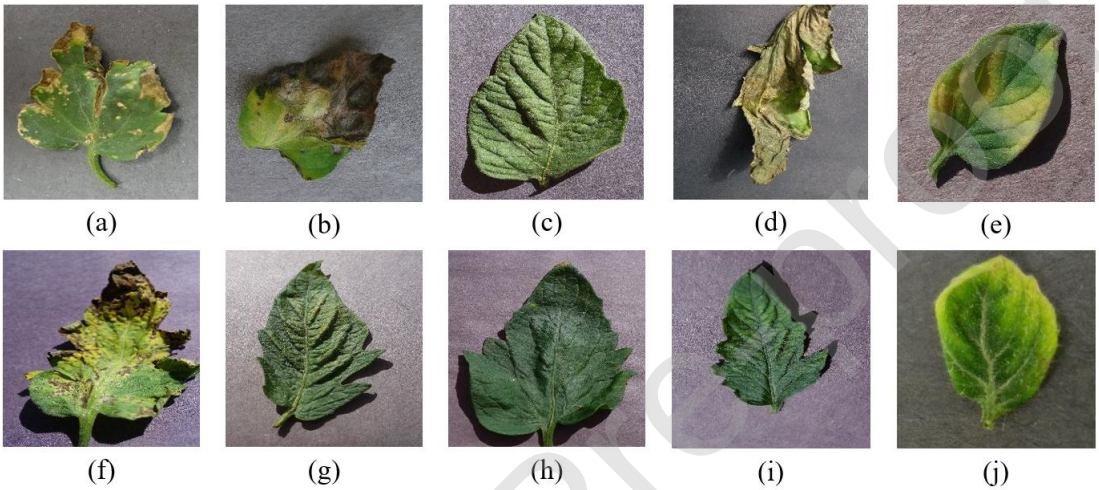

Figure 2: Sample images of different leaves affected with different diseases in tomato plant, a) Bacterial spot, b) Early blight, c) healthy, d) Late blight, e) Leaf Mold, f) Septoria leaf spot, g) Two-spotted spider mite, h) Target Spot, i) Tomato mosaic virus, j) Yellow Leaf Curl Virus.

To ensure class balance for our study, data augmentation has been carried out to ensure that each class had around 1400 images. The standard augmentation process used by researchers to increase data size is to rotate the images by 90°, 180° or 270° or take mirror reflections. For leaf data, this kind of augmentation results in the generation of images that are very similar to original images. This does not help in improving the classification accuracy. To make realistic augmentation, new images are created in this study by varying the image intensity over part of the image randomly in the range of 20 to 30, besides carrying out standard augmentation steps.

5. Results and analysis

This section describes the experiments performed on the tomato crop data of the PlantVillage dataset. The proposed 8 layer CNN model has been trained with 1400 leaf images of 10 classes. The model has been validated with 300 images of each class. The testing set consists of 100 images for each class.

5.1. Performance of proposed CNN model

The proposed CNN model was trained and tested with the dataset described above. The accuracy was computed using equation (17) as follows:

The accuracy obtained at various stages of epochs is shown in Fig. 3. As can be seen, the test accuracy increases with the number of epochs, and the best performance is obtained at 5000 epochs, after which performance starts falling. Maximum accuracy of 98.4% is obtained after 5000 epochs.

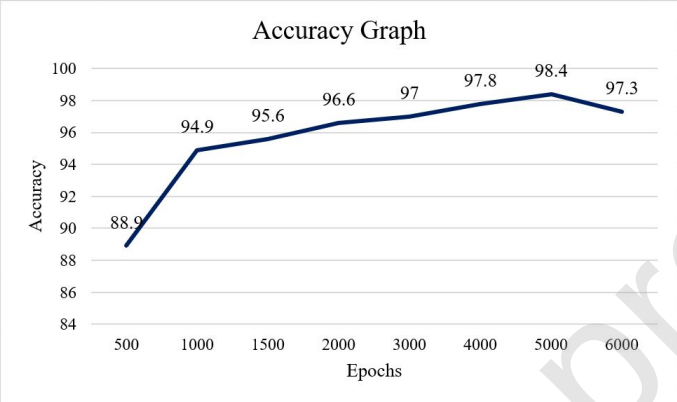

Figure 3: Curve showing the testing accuracy increase with epochs upto a certain extent.

Performance for individual disease classes is shown in the confusion matrix of Fig. 4. In this matrix, the diagonal elements (shown in blue color) depict correct predictions, while the remaining elements (shown in pink color) indicate the mis-classifications. For instance, in the first row of the confusion matrix, 99 images out of 100 test images of class 0 were correctly classified as class 0 and only 1 image was mis-classified as class 4 which is shown as error case in pink color. It is clear from the matrix, that the CNN model was able to carry out correct predictions in most of the disease classes.

The performance of the model was also tested with a different number of test images for each class. The performance of this scenario is shown in the confusion matrix of Fig. 5. For example in class 0 out of 44 instances only 1 was wrongly classified as class 4. The accuracy can be calculated by values in pink boxes divided by total instances. In this case, it was 98.64%.

5.1.1. Testing the proposed CNN model for other Domains

The proposed CNN model has been tested on datasets of other domains, such as Lungs images with 3 classes: healthy, single nodule, and multinodule, and accuracy of 99% was obtained. For these images publicly available dataset named Early Lung Cancer Action Program (ELCAP) [32] was used, which consists of Low Dose CT scans. Another dataset of Galaxy images from Kaggle [2] [33] with 3 classes: Elliptical, Spiral, Irregular was also tested and accuracy of 97

[2]Dataset can be downloaded from: https://www.kaggle.com/c/galaxy-star-separation/data

Classes	0	1	2	3	4	5	6	7	8	9
0	**99**				1					
1	1	**97**	1		1					
2	1		**97**		1	1				
3				**100**						
4		1			**98**		1			
5						**96**	4			
6			1			1	**98**			
7						1		**99**		
8									**100**	
9										**100**

Figure 4: Confusion matrix showing correct predictions in blue and wrong in red for Proposed Model on PlantVillage dataset.

Classes	0	1	2	3	4	5	6	7	8	9
0	**43**				1					
1		**67**	1							
2	1		**95**			1				
3				**37**						
4		1			**58**					
5						**75**	2			
6			1				**80**			
7						1		**93**		
8									**55**	
9										**49**

Figure 5: Confusion matrix showing correct predictions for imbalanced testing dataset from PlantVillage dataset images.

5.1.2. Performance of Variants of proposed CNN model

Experiments were also carried out on variants of the proposed low order CNN architecture, by changing the number of convolution layers, the number of hidden nodes, and using a different number of kernels. The Proposed CNN model was tested with 2, 3, and 4 convolution layers. It turned out that the 3 convolution layers CNN resulted in the highest accuracy, and therefore further experiments were done with 3 layer CNN.

The number of nodes in the penultimate Dense layer was also varied, and performance was tested for 64, 128, 256, and 512 nodes in this layer. It was found that the model performs optimally with 128 nodes, hence 128 nodes were kept in the proposed model design.

5.2. Validation and Comparison of proposed CNN with Traditional Machine Learning models

The working of the proposed model was compared with other traditional Machine Learning algorithms on the same dataset, and the results are presented in Table 3. As explained in section 2.3 the performance of the proposed model is compared with ML methods based on Accuracy (refer equation 17) and F1-score (refer equation 18) [29].

Table 3 shows the Accuracy and F1-Score of the proposed CNN model is much higher than the accuracies obtained using other ML algorithms.

Table 3: Accuracy comparison with ML models.

Features	Model	True Positive	False Positive	True Negative	False Negative	Accuracy (%)	F1-Score
Haralick Hu-features HSV LBP	SVM	99	1	621	279	72	0.4142
	Decision Tree	94	6	640	260	73.4	0.4140
	LR	99	1	798	102	89.7	0.6578
	k-NN	100	0	841	59	94.1	0.7722
	Naïve Bayes	1	99	303	597	30.4	0.0028
Haralick Hu-features HSV	SVM	96	4	632	268	72.8	0.4137
	Decision Tree	90	10	661	239	75.1	0.4195
	LR	100	0	797	103	89.7	0.6600
	k-NN	100	0	849	51	**94.9**	0.7968
	Naïve Bayes	1	99	303	597	30.4	0.0028
Haralick Hu-features	SVM	84	16	293	607	37.7	0.2123
	Decision Tree	79	21	176	724	25.5	0.1749
	LR	86	14	267	633	35.3	0.2100
	k-NN	87	13	307	593	39.4	0.2230
	Naïve Bayes	100	0	200	700	30	0.2222
Hu-features HSV	SVM	89	11	593	307	68.2	0.3588
	Decision Tree	43	57	659	241	70.2	0.2239
	LR	98	2	785	115	88.3	0.6261
	k-NN	100	0	837	63	93.7	0.7604
	Naïve Bayes	91	9	303	597	39.4	0.2309
	Proposed Model	100	0	884	16	**98.4**	**0.9259**

5.2.1. Analysis of performance of the proposed CNN model:

As explained in section 2.1 all parameters in equations of traditional Machine Learning ways are fixed for a given training feature dataset. Whereas in Deep CNN the filters are decided on run time and loss minimization equations depend on the number of input layers, input nodes, regularization, drop out, etc. which gives rise to as many variations of equations possible and with this slow combination of hyper parameters a state is reached where equation works best and gives the desired accuracy.

Fig. 6 shows the ROC curves of different ML models and the proposed CNN model. To generate a ROC curve for multiple classes, one class needs to be taken as a positive class while all the remaining classes are treated as negative class. In Fig. 6 Tomato mosaic virus class has been taken as a positive class. It is observed that the best AUC (area under the curve) belongs to the proposed model, while the other AUC values are much below 1.

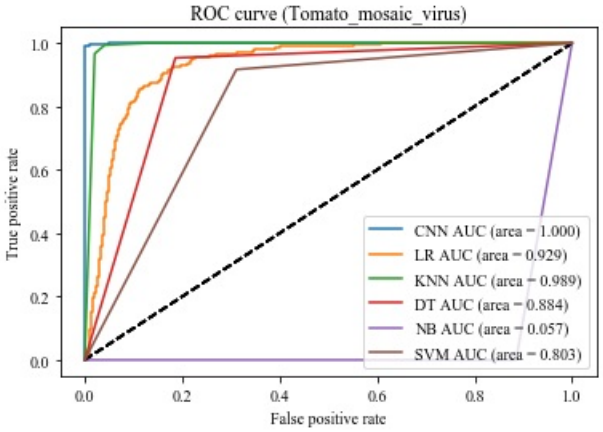

Figure 6: ROC curve comparison of Proposed Model with other ML techniques.

The reason for the CNN model performing better than traditional ML techniques is the well-proven fact that when the data size is small, the ML techniques perform well, and accuracy increase only to an extent with an increase in data, and then it starts saturating. On the other hand, the CNN model performance keeps improving as data size increases and does not have a saturation level. Hence, the proposed CNN model qualifies to be the best for plant disease classification from leaf images.

5.2.2. Comparison of accuracy of proposed CNN with Feed Forward ANN

A Feed-Forward Artificial Neural Network with 3 hidden layers was tried with the PlantVillage dataset. The input to ANN was the same features used in traditional Machine Learning methods such as Haralick features, Hu features, HSV features, and LBP features. Testing accuracy is shown in Table 4. It is seen that maximum accuracy of 92.9% of ANN is much lower than 98.4% accuracy of the proposed CNN model.

Table 4: Accuracy comparison with ANN.

Features	True Positive	False Positive	True Negative	False Negative	Accuracy (%)	F1-Score
Haralick, Hu, HSV, LBP	97	3	829	71	92.6	0.7238
Haralick, Hu, HSV	96	4	828	72	92.4	0.7164
Haralick, Hu	64	36	332	568	39.6	0.1748
Hu-moments, HSV	94	6	835	65	**92.9**	0.7258

5.3. Methods used to increase the model accuracy

Initially, after creating the dataset with simple augmentation the proposed model gave an accuracy of 91.2%. These images had a varied background. In the next step, the background was made black and this increased the accuracy to 96.1%. Next brightness of the overall

image was increased randomly to reflect the actual condition in the field, where sunlight may be falling on part of the leaves. The intensity was increased by 20-30 and on random width (20-80%) of image. The sample pre-processed images by this mechanism are shown below in Fig. 7. This image pre-processing greatly helped to increase the accuracy to 98.4%.

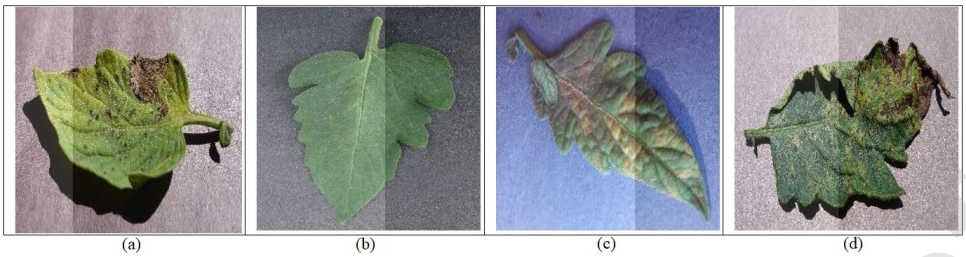

Figure 7: Random brightness increase on random width on left side of image.

5.3.1. Testing on altogether different dataset:

The proposed trained model was also tested on labeled images from a different dataset [3]. It contains labeled tomato plant leaves with 5 different classes as shown in Table 5. Each folder contains 350-600 images of the corresponding class. The accuracy of 5 classes on different models trained earlier on the PlantVillage dataset is tabulated in Table 5.

Table 5: Accuracy obtained with other dataset

S.No.	Disease Class	Mobilenet	VGG 16	InceptionV3	**Proposed Model**
1	BacterialSpot	58.135	97.627	77.288	**99.322**
2	Lateblight	98.983	98.983	96.610	**98.135**
3	SeptoriaLeafSpot	91.186	98.984	86.632	**99.492**
4	Tomato_mosaic	100.000	97.311	16.666	**98.387**
5	Yellowcurved	89.185	97.748	92.521	**98.166**

As seen from Table 5 the accuracy is quite high even though the training was done on the PlantVillage dataset. The proposed model fares much better than other models. Thus it can be assured that the best results will come when a farmer tries the application to figure out the disease for a new test leaf image.

5.3.2. Testing on actual farm images:

To further test the proposed CNN model, a visit was made to a nearby tomato farm, and images were collected for healthy and diseased plants. Some sample images are shown in Fig. 8 which were collected using a 2M pixel camera [4]. It was known that the part of the tomato crop was suffering from Late Blight disease. 50 healthy samples and 50 diseased

[3]Dataset can be downloaded from: https://github.com/PrajwalaTM/tomato-leaf-disease-detection
[4]Dataset can be downloaded from: https://github.com/mohit-aren/CNN_Tomato_dataset

samples were selected for testing on the proposed model (trained on PlantVillage dataset). The accuracy turned out to be 86.27%. The reason for slightly lower accuracy can be attributed to the fact that the tomato crop was a local one and not the same as used in the PlantVillage dataset.

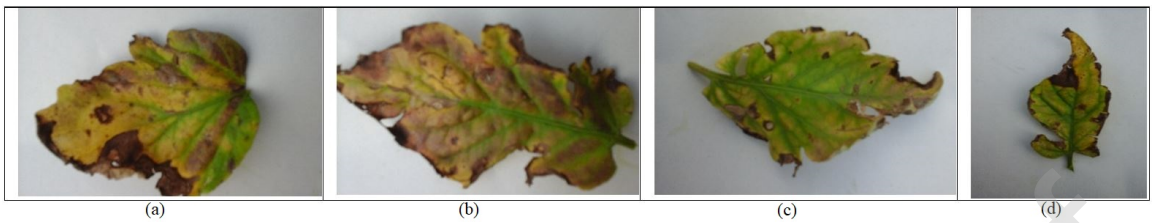

Figure 8: Image of infected leaf images from farm.

6. Discussion

Traditional Machine Learning methods have been used for quite some time for plant disease identification. With the advent of Deep Learning techniques, several well-established architectures like VGG16 [16], AlexNet [11], GoogleNet [12], InceptionV3 [30] and Resenet [17] were reported in the literature for handling a large number of image classification tasks. The success of these models inspired many researchers to use these pre-trained models for application in plant disease identification [6, 10, 14]. As these architectures were designed for the identification of 1000 classes of the Imagenet dataset, the number of convolution layers was large (typically in the range 16-40), with each layer having a large number of convolution filters (in the range 128-512). While applications involving deep learning pre-trained models achieve higher accuracy, they do so at the cost of huge storage space and large computational time for execution. For actual deployment in a field, where it would be handier to port the solution on mobile devices, this could be a serious handicap as pre-trained models usually need nearly 100 MB space. To mitigate such issues, the current paper proposes a lightweight convolution network (CNN model) that has been developed from scratch with a very small number of convolution layers and trained for a large number of epochs.

As shown in Table 6, the performance of the proposed model turns out to be superior in terms of accuracy, storage space, and execution time. The reasons can be attributed to the following:

A higher number of layers imply creating a polynomial of higher degree for a lesser number of dependent variables. The pre-trained models were developed to handle 1000 image classes, but as a number of classes in plant disease classification are much smaller, mostly in the range 4-40, the use of pre-trained models may lead to overfitting and produce erroneous results. Hence the superior accuracy of the proposed model can be attributed to its small number of layers. Many architectures were tried by varying the number of layers, and a number of filters to come up with the proposed CNN model. The robustness of the model was established by training it on PlantVillage dataset and testing on an altogether different dataset as shown in Table 5. It was also shown that a 3 Layer Artificial Neural

Network ANN model has higher accuracy than that can be obtained from traditional machine learning methods, although even this accuracy is less than the proposed CNN model.

The storage requirement of the proposed model is much smaller compared to the other pre-trained models, the reason being that the proposed 3 layer model with 32 filters needs very few weight values to be stored, while other pre-trained models need millions of weights because of 30 layers and more than 500 filters. For the same reason, the execution time of the proposed model is much smaller than the pre-trained models.

With the available dataset of PlantVillage, the maximum accuracy of the proposed model turned out to be 91.2%. However, when the data set was augmented with flipping, rotation, and random variation of leaf intensities (refer section 5.3), higher accuracy of 98.4% was achieved.

The model may have limitations, when the leaves overlap and are not on a plain background as the dataset used in the proposed work, is of PlantVillage in which leaves are given on a gray background. In other crops, these images need to be carefully taken by keeping a white/gray paper behind the leaves for best performance, else the background with stems, other leaves, etc. may need proper segmentation to achieve reasonable accuracy. The model may also have a limitation when multiple diseases may be affecting a plant and proposed model will be able to tell the most probable disease only. The detailed comparison of proposed work with state of art methods using pre-trained CNN models is given in the following subsection:

6.1. Comparison of accuracy with other Pretrained CNN models

Table 6 presents a performance comparison of the proposed model with pre-trained models on 3 popular CNN architectures, namely, VGG16 [16], Inception V3 [30], and Mobilenet [31]. It may be noted that the best validation accuracy on LeNet CNN for a similar dataset was 94.8% [34].

VGG16: VGG 16 was initially loaded with pre-trained imagenet weights and an outputt layer of 10 nodes was added corresponding to 10 classes of tomato. It was found that transfer learning using VGG16 was not very encouraging in our modeled dataset of tomato leaves from PlantVillage dataset folders. The final accuracy of 93.5% was obtained on training VGG16 code on the NVIDIA supercomputer.

Inception V3: Inception V3 pre-trained model was also used to perform transfer learning on the 10-class tomato disease problem. It was found the accuracy is even less than VGG16 and comes to 77.5% when trained on NVIDIA. It was felt that the model will work well on a greater number of classes as 42-layer deep architecture makes it overfit for a small number of classes when the distinctive features are not obvious or large.

Mobilenet: Transfer learning was used with Mobilenet architecture by adding an output layer of 10 nodes corresponding to 10 classes of tomato. The accuracy obtained with Mobilenet was 82.6%. The overfitting may be attributed to 27 convolution layers of this architecture.

The accuracy of pretrained models in prior research work has been compared with proposed CNN model in Table 6.

Table 6: Accuracy comparison with Pretrained models in prior research.

Study	Year	Objective	Images	Methods	Accuracy (%)
Too et al. [35]	2019	Plant leaf disease	54,306	VGG16	81.83
Gensheng et al. [36]	2019	Tea leaf disease	4,980	VGG16	90
Wang et al. [18]	2017	Plant leaf disease	54,306	VGG16	90.4
This study	2020	Tomato leaf disease	18,160	VGG16	93.5
Wang et al. [18]	2017	Plant leaf disease	54,306	Inception-V3	80
Gandhi et al. [37]	2018	Plant leaf disease	56,000	Inception-V3	88.6
This study	2020	Tomato leaf disease	18,160	Inception-V3	77.5
Elhassouny & Smarandache [38]	2019	Tomato leaf disease	7,176	Mobilenet	88.4
Gandhi et al. [37]	2018	Plant leaf disease	56,000	Mobilenet	92
This study	2020	Tomato leaf disease	18,160	Mobilenet	82.6
Proposed Model	**2020**	**Tomato leaf disease**	**18,160**	**CNN**	**98.4**

7. Conclusions

In this paper, a lightweight Convolution Neural Network model has been proposed for the classification of 9 types of diseases in tomato crops. For the experimental purpose, data has been taken from the PlantVillage dataset. Moreover, the number of samples in different classes vary between 200-1400. So, after applying the data augmentation techniques effort is made to balance the samples in each class. For the performance evaluation purpose traditional Machine Learning approaches along with popular Neural Network pre-trained models have been executed. The experimental results show the efficiency of the proposed algorithm is better than traditional ML approaches and pre-trained models. The performance of the proposed algorithm is compared on various evaluation metrics such as - confusion matrix, accuracy, precision, recall, F1-score, etc. The accuracy of the proposed algorithm is 98.4% which is comparable to the state of the art accuracy in leaf-based classification. The findings in this research could be applied to other crops other than tomato and also other biological domains like X-Ray images (e.g. of lungs, liver or kidneys), CT-scan images of the brain, etc. to know the disease using image classification and help the patients with low cost and fast results. As a future work, the objective is to detect the severity of the disease in tomato crops. Proposed work can also be extended to indicate the extent to which the disease has spread in the plant. This research may help in working out the amount of fungicide needed at any specific stage of plant growth.

References

[1] M. Ebrahimi, M. Khoshtaghaza, S. Minaei, B. Jamshidi, Vision-based pest detection based on svm classification method, Computers and Electronics in Agriculture 137 (2017) 52–58.

[2] A. A. Bharate, M. Shirdhonkar, A review on plant disease detection using image processing, in: 2017 International Conference on Intelligent Sustainable Systems (ICISS), IEEE, 2017, pp. 103–109.

[3] U. Mokhtar, M. A. Ali, A. E. Hassanien, H. Hefny, Identifying two of tomatoes leaf viruses using support vector machine, in: Information Systems Design and Intelligent Applications, Springer, 2015, pp. 771–782.

[4] T. Rumpf, A.-K. Mahlein, U. Steiner, E.-C. Oerke, H.-W. Dehne, L. Plümer, Early detection and classification of plant diseases with support vector machines based on hyperspectral reflectance, Computers and electronics in agriculture 74 (1) (2010) 91–99.

[5] A. Johannes, A. Picon, A. Alvarez-Gila, J. Echazarra, S. Rodriguez-Vaamonde, A. D. Navajas, A. Ortiz-Barredo, Automatic plant disease diagnosis using mobile capture devices, applied on a wheat use case, Computers and electronics in agriculture 138 (2017) 200–209.

[6] S. P. Mohanty, D. P. Hughes, M. Salathé, Using deep learning for image-based plant disease detection, Frontiers in plant science 7 (2016) 1419.

[7] G. L. Grinblat, L. C. Uzal, M. G. Larese, P. M. Granitto, Deep learning for plant identification using vein morphological patterns, Computers and Electronics in Agriculture 127 (2016) 418–424.

[8] X. Yang, T. Guo, Machine learning in plant disease research, European Journal of BioMedical Research 3 (1) (2017) 6–9.

[9] A. K. Rangarajan, R. Purushothaman, A. Ramesh, Tomato crop disease classification using pre-trained deep learning algorithm, Procedia computer science 133 (2018) 1040–1047.

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

[11] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.

[13] D. Hughes, M. Salathé, et al., An open access repository of images on plant health to enable the development of mobile disease diagnostics, arXiv preprint arXiv:1511.08060 (2015).

[14] X. Zhang, Y. Qiao, F. Meng, C. Fan, M. Zhang, Identification of maize leaf diseases using improved deep convolutional neural networks, IEEE Access 6 (2018) 30370–30377.

[15] A. Fuentes, S. Yoon, S. Kim, D. Park, A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition, Sensors 17 (9) (2017) 2022.

[16] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).

[17] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[18] G. Wang, Y. Sun, J. Wang, Automatic image-based plant disease severity estimation using deep learning, Computational intelligence and neuroscience 2017 (2017).

[19] A. Khamparia, A. Singh, A. K. Luhach, B. Pandey, D. K. Pandey, Classification and identification of primitive kharif crops using supervised deep convolutional networks, Sustainable Computing: Informatics and Systems (2019).

[20] G. Hu, X. Yang, Y. Zhang, M. Wan, Identification of tea leaf diseases by using an improved deep convolutional neural network, Sustainable Computing: Informatics and Systems 24 (2019) 100353.

[21] S. S. Haykin, et al., Neural networks and learning machines/simon haykin. (2009).

[22] A. Y. Ng, M. I. Jordan, On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes, in: Advances in neural information processing systems, 2002, pp. 841–848.

[23] G. Guo, H. Wang, D. Bell, Y. Bi, K. Greer, Knn model-based approach in classification, in: OTM

Confederated International Conferences" On the Move to Meaningful Internet Systems", Springer, 2003, pp. 986–996.
[24] J. R. Quinlan, Induction of decision trees, Machine learning 1 (1) (1986) 81–106.
[25] C.-Y. J. Peng, K. L. Lee, G. M. Ingersoll, An introduction to logistic regression analysis and reporting, The journal of educational research 96 (1) (2002) 3–14.
[26] E. Maggiori, Y. Tarabalka, G. Charpiat, P. Alliez, Convolutional neural networks for large-scale remote-sensing image classification, IEEE Transactions on Geoscience and Remote Sensing 55 (2) (2017) 645–657.
[27] M. Agarwal, A. Sinha, S. K. Gupta, D. Mishra, R. Mishra, Potato crop disease classification using convolutional neural network, in: Smart Systems and IoT: Innovations in Computing, Springer, 2020, pp. 391–400.
[28] M. Sokolova, G. Lapalme, A systematic analysis of performance measures for classification tasks, Information processing & management 45 (4) (2009) 427–437.
[29] A. Tharwat, Classification assessment methods, Applied Computing and Informatics (2018).
[30] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2818–2826.
[31] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, arXiv preprint arXiv:1704.04861 (2017).
[32] S. Raghava, S. Siddque, Pub075 survival of patients with stage i lung cancer detected on ct screening in south indian population, Journal of Thoracic Oncology 12 (1) (2017) S1491–S1492.
[33] D. Misra, S. N. Mohanty, M. Agarwal, S. K. Gupta, Convoluted cosmos: Classifying galaxy images using deep learning, in: Data Management, Analytics and Innovation, Springer, 2020, pp. 569–579.
[34] P. Tm, A. Pranathi, K. SaiAshritha, N. B. Chittaragi, S. G. Koolagudi, Tomato leaf disease detection using convolutional neural networks, in: 2018 Eleventh International Conference on Contemporary Computing (IC3), IEEE, 2018, pp. 1–5.
[35] E. C. Too, L. Yujian, S. Njuki, L. Yingchun, A comparative study of fine-tuning deep learning models for plant disease identification, Computers and Electronics in Agriculture 161 (2019) 272–279.
[36] G. Hu, H. Wu, Y. Zhang, M. Wan, A low shot learning method for tea leaf's disease identification, Computers and Electronics in Agriculture 163 (2019) 104852.
[37] R. Gandhi, S. Nimbalkar, N. Yelamanchili, S. Ponkshe, Plant disease detection using cnns and gans as an augmentative approach, in: 2018 IEEE International Conference on Innovative Research and Development (ICIRD), IEEE, 2018, pp. 1–5.
[38] A. Elhassouny, F. Smarandache, Smart mobile application to recognize tomato leaf diseases using convolutional neural networks, in: 2019 International Conference of Computer Science and Renewable Energies (ICCSRE), IEEE, 2019, pp. 1–4.
[39] M. T. Shakoor, K. Rahman, S. N. Rayta, A. Chakrabarty, Agricultural production output prediction using supervised machine learning techniques, in: 2017 1st International Conference on Next Generation Computing Applications (NextComp), IEEE, 2017, pp. 182–187.
[40] D. Han, Q. Liu, W. Fan, A new image classification method using cnn transfer learning and web data augmentation, Expert Systems with Applications 95 (2018) 43–56.
[41] G. P. Zhang, Neural networks for classification: a survey, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 30 (4) (2000) 451–462.

Appendix A.

Appendix A.1. Visualization of hidden layers output:

The hidden layers output can be seen in Fig. A.9, A.10 and A.11.

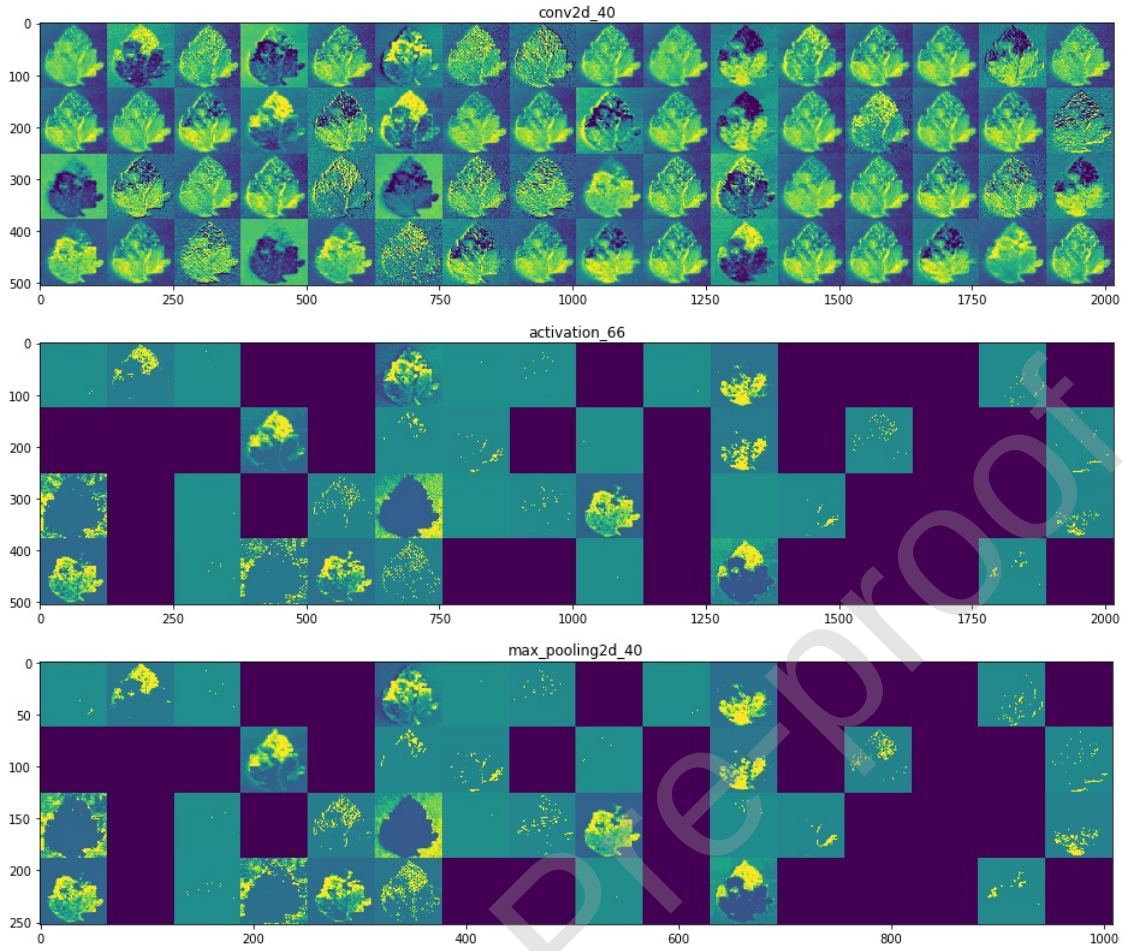

Figure A.9: (a) 1st Convolution Layer Output (b) After Relu activation (c) After Max Pooling.

As seen in Fig. A.9, initially 1st layer with 64 filters show 64 images showing shape of leaf. After ReLU activation some images become dark due to negative weights made zero. The edges and different features of leaf starts showing clearly here.

As seen in Fig. A.10, 2nd layer with 32 filter show 32 images in which further complex features are getting deciphered. These images are after training for 5000 epochs and loading the trained weight file and then checking activation images for a particular leaf.

As seen in Fig. A.11, 3rd layer with 8 filters show 8 images. The leaf representation is clearer in initial layer and subsequently difficult to visualize as more complex features such angles, single edges and corners are deciphered. This 2D matrix is later flattened and passed to Dense layer and then final softmax layer.

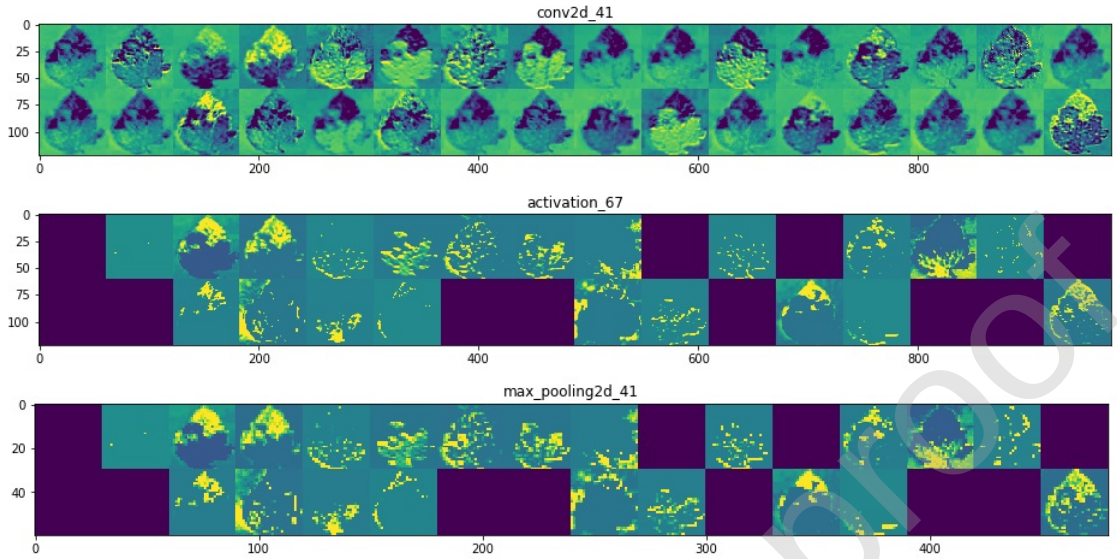

Figure A.10: (a) 2nd Convolution Layer Output (b) After Relu activation (c) After Max Pooling.

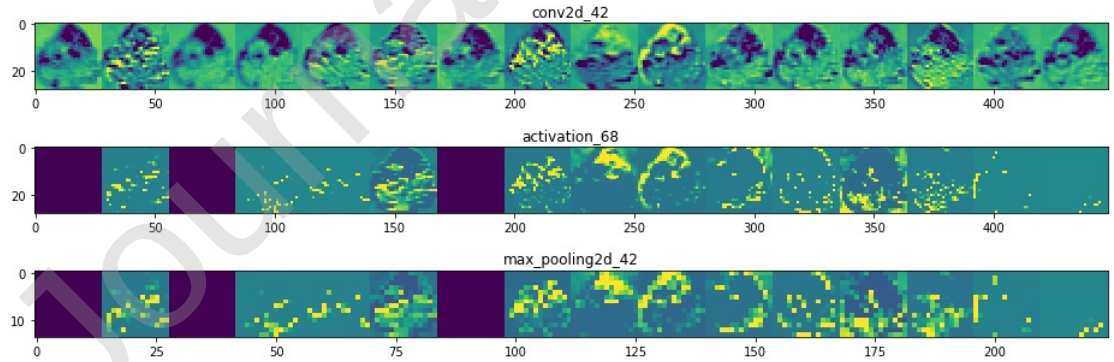

Figure A.11: (a) 3rd Convolution Layer Output (b) After Relu activation (c) After Max Pooling.