Smart Agricultural Technology 5 (2023) 100301

Contents lists available at ScienceDirect

Smart Agricultural Technology

journal homepage: www.journals.elsevier.com/smart-agricultural-technology

Detecting tomato leaf diseases by image processing through deep convolutional neural networks

Md. Iqbal Hossain [a], Sohely Jahan [a,*], Md. Rashid Al Asif [a], Md. Samsuddoha [a], Kawsar Ahmed [b,c,*]

[a] Department of Computer Science and Engineering, University of Barishal, Barishal-8254, Bangladesh
[b] Group of Biophotomatiχ, Department of Information and Communication Technology, Mawlana Bhashani Science and Technology University, Santosh, Tangail-1902, Bangladesh
[c] Health Informatics Research Lab, Department of Computer Science and Engineering, Daffodil International University, Daffodil Smart City, Birulia, Dhaka-1216, Bangladesh

ARTICLE INFO

Editor: Stephen Symons

Keywords:
Leaf disease detection
Deep learning
Neural networks
Color models
Median filter
Gaussian filter

ABSTRACT

Machine Learning (ML) and Deep Learning (DL) have already brought unprecedented success in the detection of various diseases of plant leaves, fruits, buds, flowers, etc. Besides, computer science and related field researchers are widely trying to use specific ML and DL methods to classify images and get better results in the field of agriculture and technology. Considering these, Deep Convolutional Neural Networks (DCNN) have been applied in this research. We first applied the Gaussian filter and the Median filter separately on the main dataset and saved the filtered images into two separate directories. We then applied two color models (HSI and CMYK) separately to the images in each directory. Thus, we pre-processed the images in four different ways with the main objective of finding the best combination of the filtering methods and the color models. We then applied our selected DCNN models to each output obtained from the pre-processing steps and finally chose the best methodology based on the accuracy. At last, we have found the highest accuracies (98.27% in Vgg-19, 94.98% in MobileNet-V2, and 99.53% in the ResNet-50) by using the Gaussian Blur and the Gaussian Noise filters with the RGB to CMYK color conversion method.

1. Introduction

The growth of agriculture has been intimately tied to the advancement of science and computer technology and also artificial intelligence (AI) has emerged as a viable technique to enhance agricultural outcomes by offering insightful advice and recommendations regarding crops. Besides, agricultural crops or vegetables are used in almost every country in the world and meet our needs in many ways. Tomato ***(Solanum Lycopersicum L.)*** [1] is one of the most popular, the second most important fruit or vegetable crop next to potato ***(Solanum tuberosum L.)*** [2]. Tomatoes are loved all over the world as a vegetable, as a sauce, as a salad, and even as a natural skin care product. However, many diseases in the leaves of tomatoes greatly reduce the fruit of the tomato and sometimes even kill the plant. Some common diseases are Septoria Leaf Spot, Early Blight, Late Blight, Tomato Yellow Leaf Curl Virus, etc. These diseases can be detected manually or using technology, and it is possible to multiply the yield of tomatoes and meet the needs of the world better.

However, due to issues like ignorance, expensive expenses, or a lack of understanding of the advantages of cutting-edge technologies, conventional farming practices are still widely used in many nations. This ignorance causes issues and heightens the difficulties farmers already face. Moreover, it takes a lot of time to diagnose the disease and take appropriate action against it manually. That is why researchers are trying to apply different branches of AI such as ML, DL, Computer Vision (CV), various Neural Networks (NN), etc., in the fields of agriculture. Therefore, three common diseases (Septoria Leaf Spot, Early Blight, Tomato Yellow Leaf Curl Virus) of tomatoes have been identified in this research. By applying this, we can help farmers to take major steps to increase tomato production around the world.

In our work, we prioritized DCNN over other DL methods for our tomato leaf disease detection system to achieve higher accuracy and improved efficiency. Because DCNN provides many additional advan-

* Corresponding authors.
E-mail addresses: iqbal.cse4.bu@gmail.com (M.I. Hossain), sohely.cse@gmail.com (S. Jahan), rashid.al.asif@gmail.com (M.R. Al Asif), sams.csebu@gmail.com (M. Samsuddoha), kawsar.ict@mbstu.ac.bd (K. Ahmed).

https://doi.org/10.1016/j.atech.2023.100301
Received 25 June 2023; Received in revised form 8 August 2023; Accepted 12 August 2023
Available online 22 August 2023

tages in image feature detection than other methods. First of all, DCNN is built in such a way that it can automatically capture image features. It also performs well in edge detection and low-level feature detection and combines them to generate more accurate results. Another important feature of DCNN is that it does not perform position-based image detection. This is very important because the leaf disease symptoms are located in different places on each leaf, which is somewhat challenging to detect with other methods. Additionally, DCNN facilitates transfer learning methods as it is built with pre-trained on very large data (ImageNet). By using this feature, it can work very easily and smoothly on a dataset like our tomato leaf disease dataset. Moreover, one of the most important features of DCNN is its ability to operate on different variations of input data such as different color variations of the image data, different noise associated with the image, and environmental variations which are very common in real-world data.

The main novelty of this research is the approaches and the way we have found the best performance to detect the selected tomato leaf diseases despite the presence of noises, color variations, and environmental differences. We have combined the color models and the filtering methods in many ways and applied them to our dataset to make our images more noise-free and more smooth so that the feature extraction becomes better for our DCNN models. Finally, we have fitted the outputs which have been obtained from our pre-processing steps to our selected DCNN models and showed the final results in the form of classification reports and confusion matrices for each outcome.

It was John McCarthy who introduced “Artificial Intelligence” to everyone in the world at the 1955 Dartmouth Conference [3]. Since then, scientists have been trying to figure out how to apply it in almost all sectors of the world.

The application of AI with the expert system in agriculture was first attempted [4] by McKinion and Lemmon in 1985 [5]. Both McKinion and Lemmon later did more important research on the application of AI to agriculture. In 1986, Lemmon proposed an expert system called COMAX (Cotton Management Expert). Later on, he also developed a computer model called GOSSYM. And for the very first time, the progress of cotton crops was effectively simulated using an expert system (COMAX) that was successfully combined with a computerized model (GOSSYM) [6].

Since then, the use of this technology in agriculture has progressively increased, and the development of new computational methods has sped up this process. In spite of the limited number of case studies, algorithms like CNN (Convolutional Neural Networks) and DBN (Deep Belief Networks) show promise for processing vast amounts of complex data in the future [7].

After that, a considerable amount of work has already been done by applying CNN to diagnose tomato leaves. One of the most recent works is Kibriya et al. [8] which was published in IEEE in 2021. In their proposed method, two different transfer learning models, i.e., VGG16 and GoogleNet are trained. Then the deep feature vector is supplied to other classifiers like Cubic SVM, Ensemble Bagged Tree, Fine KNN, and Medium Gaussian SVM. They work with three types of tomato leaf diseases such as Bacterial Spot, Late Blight, and Early Blight. Methods of pre-processing images are not in detail in their work. Also, The classification of the images and the segmentation methods are not clear in their paper. The comparison of different types of methods needs to be more realistic and informative along with the performance evaluation of the CNN models.

In 2020 Jiang et al. [9] proposed a Resnet-50 residual model to detect several leaf diseases of Tomatoes. They trained the model efficiently and predicted the result with reasonable accuracy. Their model’s train accuracy is 98.3% and the test accuracy is 98%. For extracting the feature, they use a 7x7 convolution kernel size in the initial stages. After that, they use an 11x11 convolution kernel which gives them 2.3% increased accuracy in comparison with the first kernel. They also use the Leaky-ReLU activation function for generating and comparing the result. For preventing overfitting issues, they have applied random data augmentation to their model. 3000 images have been used in that model. The images were expanded further by Gaussian noise and Affine transformation which sums up a total of 6794 images. There is a lack of a relatively complete and high-quality crop disease database, the classification of some rare diseases or species in their paper.

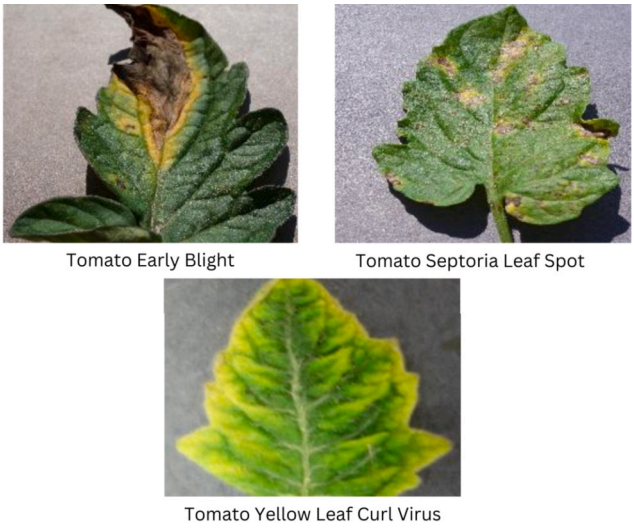


Fig. 1. Images of our selected 3 leaf diseases of tomatoes.

Also in 2020, Asok et al. [10] proposed a CNN model to detect four types of tomato leaf conditions such as Healthy, Phoma Rot, Leaf Miner, and Target Spot. In the preprocessing steps, they applied the Gaussian filter to reduce the noise of the images. DWT (Discrete Wavelet Transform) and GLCM (Gray Level Co-occurrence Matrix) are applied in feature extraction steps. DWT is used to improve the enhancement of the images which can take the discrete wavelength transform of the images and give equal precision. GCLM is used to classify the images into different luminous levels. Their proposed CNN model gives 98.12% accuracy.

Tm et al. [11] have experimented with several standard deep learning architectures like AlexNet, GoogleNet, and LeNet. They found the best result with LeNet as LeNet is a type of CNN model that is a combination of convolution, activation, pooling, and fully connected layers. They also use an additional layer in their proposed LeNet model than the original LeNet model. They have used a minimum number of layers to detect the diseases, not the deep layers. Also lack of different learning rates and optimizations that could be used in future work.

Kumar et al. [12] trained the DCNN models with all 14,903 images of the 9 classes from the “Plant Village”. LeNet, VGG16, Xception, and Resnet50 have been trained in their research. Softmax classifiers have been used for the fully connected layers. Data augmentation, regularization, and dropout (to avoid overfitting problems) have been used in their models.

The remaining sections are arranged as follows: Section 2 will be used to describe the Materials and Methodology. The Result and Discussion and finally the Conclusion will be described in sections 3 and 4 respectively.

2. Materials and methodology

2.1. Materials:- data source and description

Machine learning and datasets are inextricably linked. In order to train and test a machine or system using an artificial neural model, the need for a highly accurate, efficient, and relatable dataset comes first. In our research, we have used a very effective and enriched dataset from Kaggle called “Plant Village Dataset” [13] where different types of tomato leaf images with many types of diseases are present. From those, we have chosen Early Blight, Septoria Leaf Spot, and Yellow Leaf Curl Virus diseases for our research. We have used around 7,000 images of these diseases to train our models. Some images of our dataset are given in Fig. 1.

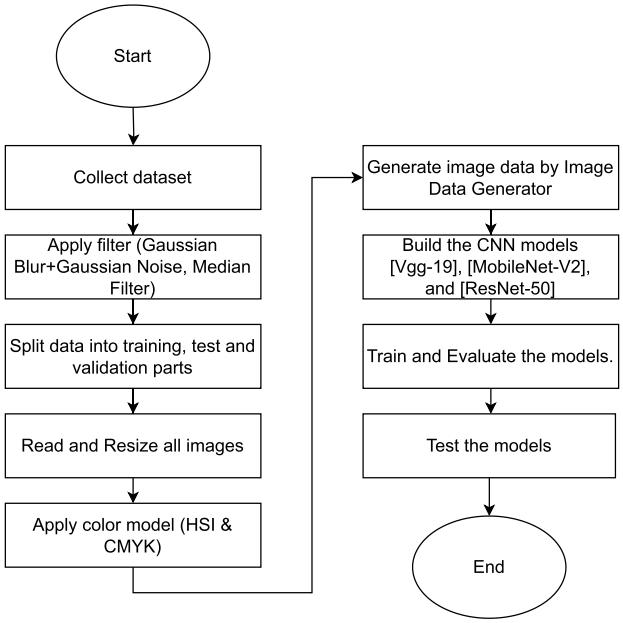


Fig. 2. Overall Methodology of our research.

2.2. Methodology

In most cases, when dealing with image processing using DCNN, we use different deep-learning models to make it happen. We have also done this. Fig. 2 describes all the processes of our research in brief.

Below are the sections that describe all the steps in detail.

2.3. Filtering in pre-processing steps

We have applied the Gaussian Noise along with the Gaussian Blur filter and the Median filter separately to our dataset for making a clear distinction between their performances on our dataset as they can extract valuable characteristics by the knowledge of artificial neural networks.

2.3.1. Gaussian blur

The Gaussian blur filter [14], also called the Gaussian filter, is a widely used image-blurring filter in image processing that helps to decrease noise and details in an image.

The equation for the Gaussian blur filter can be given as:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{1}$$

where, x and y are the pixel coordinates, σ is the standard deviation of the Gaussian distribution and $G(x,y)$ is the Gaussian kernel at pixel (x,y).

2.3.2. Gaussian noise

Gaussian noise [15] is a type of image noise that is generated from a Gaussian distribution. It can be added to an image to simulate camera noise or other types of natural noise. The equation of the Gaussian noise is:

$$I_{noisy}(x,y) = I(x,y) + \epsilon(x,y) \tag{2}$$

where $I(x,y)$ represents the original picture, $I_{noisy}(x,y)$ represents the noisy image, and $\epsilon(x,y)$ represents the Gaussian noise, which has a mean of zero and a standard deviation of σ.

2.3.3. Median filter

Median filter or Median blur [16] is used to reduce the noise in images. Each pixel's value is changed to reflect the median value of the pixels around it. Unlike other blurring methods, median blur keeps the image's edges while reducing noise.

The median blur equation can be written as follows:

$$I_{out}(x,y) = test_{median}(I_{in}(x-k : x+k, y-k : y+k)), \tag{3}$$

where k is the radius of the median filter kernel, I_{in} is the input image, I_{out} is the output image following median filtering, x and y are the coordinates of the processed pixel and $test_{median}()$ is the median function that determines the median value of the surrounding pixels.

2.4. Split data into train, validation, and test sets

In machine learning and data science, dividing data into train, validation, and test sets is a standard approach to assessing and comparing the performance of various models. The key justifications for breaking up the data into such sets are model selection, overfitting prevention, and model evaluation. We split our data into three sets; one for training our model, one for validation, and another for evaluation and testing purposes. We randomly split all our images into three files, one file with 70% of the main data as a training set, 20% in the validation set, and 10% in the test set.

2.5. Color model

The color models RGB (Red, Green, Blue), CMYK (Cyan, Magenta, Yellow, Key/Black), HSL (Hue, Saturation, Lightness), HSI (Hue, Saturation, Intensity), and others are used in image processing. We have applied HSI and CMYK color models along with the filters that are described above to find our best model in terms of performance. The two color models are described below.

2.5.1. HSI color model

We have applied the HSI color model [17] for this research. The HSI color model is frequently employed in image processing applications including image augmentation, contrast adjustment, and image segmentation. The formulas are:

$$Hue(H) = \arccos\left(\frac{0.5 * ((R-G)+(R-B))}{\sqrt{(R-G)^2+(R-B)*(G-B)}}\right) \tag{4}$$

$$Saturation(S) = 1 - 3\frac{\min(R,G,B)}{R+G+B} \tag{5}$$

$$Intensity(I) = \frac{R+G+B}{3} \tag{6}$$

where the processed pixel's red, green, and blue values are R, G, and B.

2.5.2. CMYK color model

In contrast to the RGB model, which adds light to produce colors, the CMYK model [17] subtracts light from white to produce colors. The equation for converting RGB to CMYK color model is:

$$\text{CMYK} = (C', M', Y', K) \tag{7}$$

where,

$$C' = \frac{C-K}{1-K+10^{-8}} \tag{8}$$

$$M' = \frac{M-K}{1-K+10^{-8}} \tag{9}$$

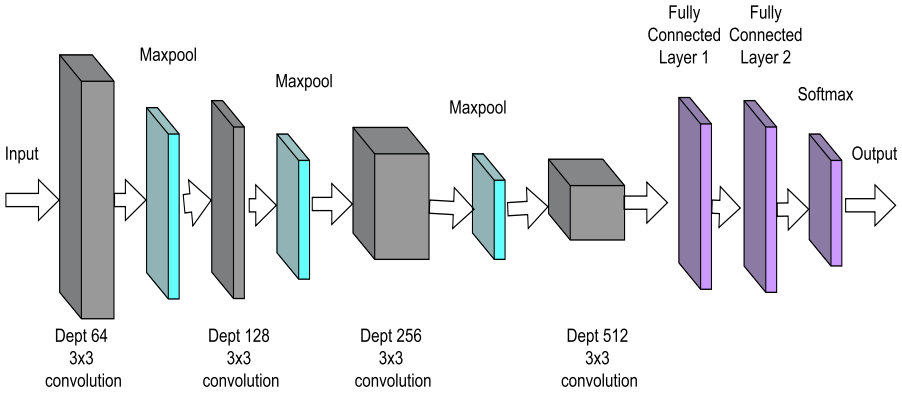
$$Y' = \frac{Y-K}{1-K+10^{-8}} \tag{10}$$

And

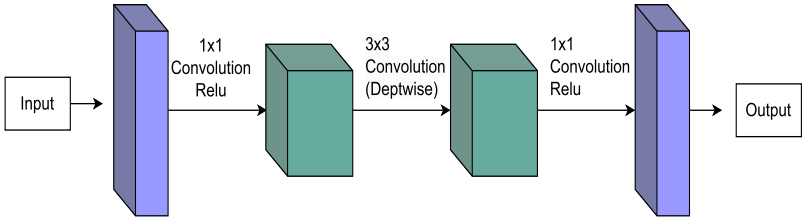
$$K = \min(\min(C,M),Y) \tag{11}$$

$$C = 1-R \tag{12}$$

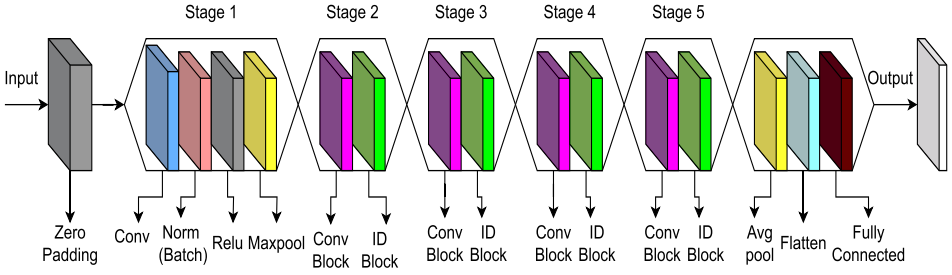
$$M = 1-G \tag{13}$$



(a) Architecture of VGG-19 model



(a) Architecture of MobileNet-V2 model



(a) Architecture of ResNet-50 model

Fig. 3. Architecture of Proposed Deep Convolutional Neural Network models.

$$Y = 1 - B \tag{14}$$

2.6. Generating image data

We have generated image data by Image Data Generator so that the machine can detect all the pixels by numerical values (between 0 to 1) and can differentiate one image's features from another and can generate a better result. We have used a variety of techniques for enhancing and modifying our images. We have used rotation, zooming, shifting, horizontal flipping, and shear range to generate data from our images.

2.7. Architecture of proposed deep convolutional neural network models

We have constructed three DCNN models for our research such as Vgg-19 [18], MobileNet-V2 [19], and ResNet-50 [20]. The architectures of the DCNN models are given in Fig. 3.

2.8. Activation: ReLU

Rectified Linear Unit (ReLU) is one of the most commonly used activation functions in DCNN. Here, we have used the flattening layer to convert the previous layer's output into a one-dimensional tensor first. Then we created a fully connected layer with 256 neurons that take the flattened output as an input. Finally, we applied the ReLU activation function to the previous layer's output. The equation [21] is:

$$f(x) = \max(0, x) \tag{15}$$

where x is the input to the function.

2.9. Activation: softmax

We have used Softmax [22] as an activation function, which is frequently used in machine learning, especially for classification problems. The Softmax function's output values range from 0 to 1, making it suitable for situations involving several classes of classification. The softmax function is applied as follows:

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} \tag{16}$$

where x_i is the i-th element of the input vector and n is the length of the input vector.

2.10. Optimizer: Adam

A CNN can employ a variety of optimizers. In this research, We have used the Adam Optimizer [23]. Because it has been demonstrated to perform well in a variety of networks with deep learning. The following are the mathematical formulas for the parameter update rule for the Adam optimizer:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{17}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{18}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{19}$$

$$\hat{v}t = \frac{v_t}{1-\beta_2^t} \tag{20}$$

$$\theta_t = \theta t - 1 - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{21}$$

Here,

- The first and second moments of the gradient at time step t are represented in this equation by m_t and v_t, respectively.
- The decay rates for the first and second moments are β_1 and β_2, correspondingly.
- The bias-corrected estimates of the first and second moments are $\hat{m}_t$ and $\hat{v}_t$, respectively.
- The current estimate of the model's parameters at time step t is θ_t.
- The learning rate is denoted by α, and a tiny constant called ϵ is used to prevent division by zero.
- The exponentiation, not the index, is shown by the superscripts on β_1 and β_2.

2.11. Loss function: sparse categorical cross-entrophy

The Sparse Categorical Cross-entropy loss function [24], [25] also known as Sparse Softmax Cross-entropy works with integer labels as opposed to categorical cross-entropy, which needs the target labels to be in one-hot encoding format. The following is the mathematical formula for the Sparse Categorical Cross-entropy.

$$L(y_{true}, y_{pred}) = -\sum_{i=1}^{C} y_{true,i} \log(y_{pred,i}) \tag{22}$$

where,

- C is the number of classes
- y_{true} is the true label vector in integer format
- y_{pred} is the predicted probability distribution vector over C classes

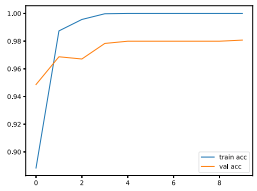
2.12. Callback [early stopping]

We have used Callback [Early Stopping] [26] to prevent overfitting issues for our models. It is used during the training of our models.

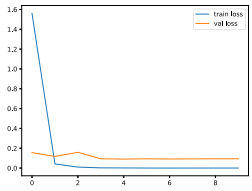
The equation is:

$$\text{EarlyStopping} = \text{argmin}_i \{\text{Loss}_i + \lambda \cdot C(i)\} \tag{23}$$

where $Loss_i$ is the loss at epoch i, C(i) denotes a function that computes the cost depending on the epoch number, and λ denotes a parameter of regularization that regulates the severity of the cost to prevent overfitting.



(a) Training and validation accuracies

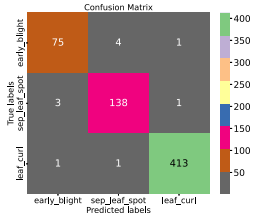


(b) Training and validation losses

	precision	recall	f1-score	support
0	0.95	0.94	0.94	80
1	0.97	0.97	0.97	142
2	1.00	1.00	1.00	415
accuracy			0.98	637
macro avg	0.97	0.97	0.97	637
weighted avg	0.98	0.98	0.98	637

Accuracy: 0.9827

(c) Classification report



(d) Confusion matrix

Fig. 4. Gaussian Blur and the Gaussian Noise filter with RGB to CMYK color conversion in VGG-19.

2.13. Confusion matrix

We have used confusion metrics [[27], [28], [29]] to show the performance of our proposed models. Several metrics are calculated including accuracy (in eq. (24)), precision (in eq. (25)), recall (in eq. (26)), and F1 score (in eq. (27)) which offers a complete perspective of our selected model's performance.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{24}$$

$$Precision = \frac{TP}{TP + FP} \tag{25}$$

$$Recall = \frac{TP}{TP + FN} \tag{26}$$

$$F1 - Score = \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{27}$$

where, TP, TN, FP, and FN represent true positives, true negatives, false positives, and false negatives respectively.

3. Result and performance discussion

We will describe the result and performance in 4 different ways that we have applied separately by the combination of the Gaussian Blur and the Gaussian Noise filter, and the Median filter along with the HSI and CMYK color models such as:

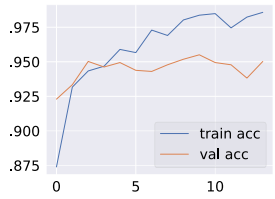
1) Gaussian Blur and the Gaussian Noise filter with RGB to CMYK color conversion
2) Gaussian Blur and the Gaussian Noise filter with RGB to HSI color conversion
3) Median filter with RGB to CMYK color conversion
4) Median filter with RGB to HSI color conversion

Finally, We have chosen our best methodology based on the performance of the three DCNN models.
The performances of all the methodologies are given below:

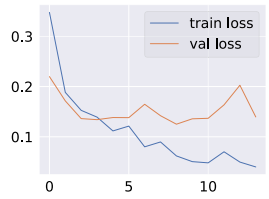
3.1. Gaussian Blur and the Gaussian Noise filter with RGB to CMYK color conversion

3.1.1. VGG-19

Fig. 4(a) is showing the training and validation accuracy of the Gaussian Blur and the Gaussian Noise filter with the RGB to CMYK



(a) Training and Validation accuracies

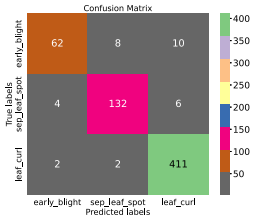


(b) Training and Validation losses

	precision	recall	f1-score	support
0	0.91	0.78	0.84	80
1	0.93	0.93	0.93	142
2	0.96	0.99	0.98	415
accuracy			0.95	637
macro avg	0.93	0.90	0.91	637
weighted avg	0.95	0.95	0.95	637

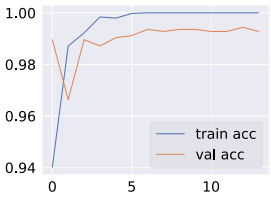
Accuracy: 0.9498

(c) Classification report

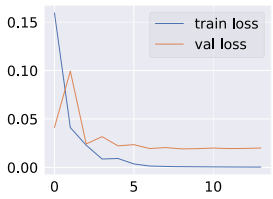


(d) Confusion matrix

Fig. 5. Gaussian Blur and the Gaussian Noise filter with RGB to CMYK color conversion in MobileNet-V2.



(a) Training and Validation accuracies

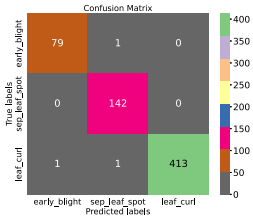


(b) Training and validation losses

	precision	recall	f1-score	support
0	0.99	0.99	0.99	80
1	0.99	1.00	0.99	142
2	1.00	1.00	1.00	415
accuracy			1.00	637
macro avg	0.99	0.99	0.99	637
weighted avg	1.00	1.00	1.00	637

Accuracy: 0.9953

(c) Classification report



(d) Confusion matrix

Fig. 6. Gaussian Blur and the Gaussian Noise filter with RGB to CMYK color conversion in ResNet-50.

color conversion method. From the graph, it is clear that, in Vgg-19, we have achieved 100% training accuracy and 98.07% validation accuracy. Besides Fig. 4(b), it is showing zero training loss and a very lower validation loss. All these data sum up that our model is well-trained and predicts the results well. In this case, the classification report containing the precision, recall, f1-score, support, macro average, weighted average, and final accuracy of the model is shown in Fig. 4(c). The confusion matrix, which represents the true and predicted levels, is given in Fig. 4(d). The final accuracy of the Vgg-19 model regarding this method is 98.27%.

3.1.2. MobileNet-V2

In MobileNet-V2, the training accuracy and the validation accuracy in Fig. 5(a) are 98.58% and 95.02% respectively. Whereas Fig. 5(b) which represents the training and validation losses, shows a training loss of 4.00% and a validation loss of 13.94%. Fig. 5(c) represents the classification report and Fig. 5(d) represents the Confusion Matrix of Gaussian Blur and the Gaussian Noise filter with RGB to CMYK color conversion methods applied in MobileNet-V2. The final accuracy of our MobileNet-V2 model is 94.98%.

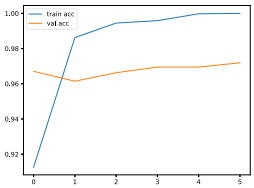
3.1.3. ResNet-50

ResNet-50 has shown 100% training accuracy and 99.28% validation accuracy (shown in Fig. 6(a)). Fig. 6(b) shows the training and validation loss. It is clear that the training loss is also zero with this method. Classification metrics for the ResNet-50 model are given in Fig. 6(c) for every tomato leaf disease class (Early Blight, Septoria Leaf Spot, and Yellow Leaf Curl Virus). The confusion matrix is given in Fig. 6(d) for this methodology. For this method, ResNet-50 has shown a test accuracy of 99.53% which is the highest accuracy among all of our selected models.

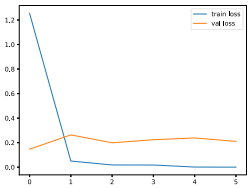
3.2. Gaussian Blur and the Gaussian Noise filter with RGB to HSI color conversion

3.2.1. VGG-19

The training and validation accuracy of Gaussian Blur and Gaussian Noise filters with the RGB to CMYK color conversion method are illustrated in Fig. 7(a). The graph clearly demonstrates that our Vgg-19 model achieves 100% training accuracy and 97.19% validation accuracy. Additionally, Fig. 7(b) displays zero training loss and 21.03% validation loss. These results provide strong evidence that our model



(a) Training and validation accuracy

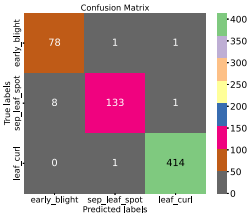


(b) Training and validation losses

	precision	recall	f1-score	support
0	0.91	0.97	0.94	80
1	0.99	0.94	0.96	142
2	1.00	1.00	1.00	415
accuracy			0.98	637
macro avg	0.96	0.97	0.97	637
weighted avg	0.98	0.98	0.98	637

Accuracy: 0.9812

(c) Classification report



(d) Confusion matrix

Fig. 7. Gaussian Blur and the Gaussian Noise filter with RGB to HSI color conversion in VGG-19.

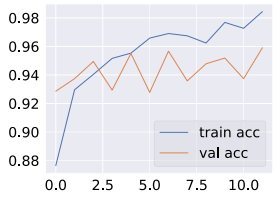
is well-trained and proficient in predicting the outcomes. Furthermore, Fig. 7(c) presents the classification report, and the corresponding confusion matrix, for the true and predicted levels, is depicted in Fig. 7(d). Based on this method, our Vgg-19 model achieves a final accuracy of 98.12%.

3.2.2. MobileNet-V2

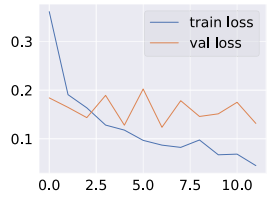
The training accuracy and validation accuracy for MobileNet-V2 are shown in Fig. 8(a) as 98.44% and 95.98%. Fig. 8(b), which illustrates the training and validation losses, is similar to our prior model (Vgg-19), with a training loss of 4.46% and a validation loss of 13.16%. Fig. 8(c) displays the classification report. Besides, Fig. 8(d) displays the confusion matrix for the Gaussian Blur and Gaussian Noise filters with RGB to CMYK color conversion methods in MobileNet-V2. Our MobileNet model's ultimate accuracy for this approach is 95.13%.

3.2.3. ResNet-50

Fig. 9 illustrates all the performance of ResNet-50, which demonstrated 100% training accuracy and 99.52% validation accuracy (shown



(a) Training and validation accuracy

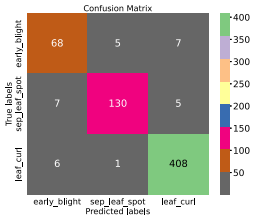


(b) Training and validation losses

	precision	recall	f1-score	support
0	0.84	0.85	0.84	80
1	0.96	0.92	0.94	142
2	0.97	0.98	0.98	415
accuracy			0.95	637
macro avg	0.92	0.92	0.92	637
weighted avg	0.95	0.95	0.95	637

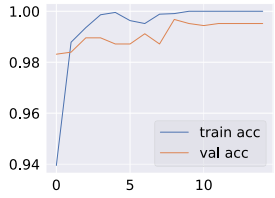
Accuracy: 0.9513

(c) Classification report



(d) Confusion matrix

Fig. 8. Gaussian Blur and the Gaussian Noise filter with RGB to HSI color conversion in MobileNet-v2.



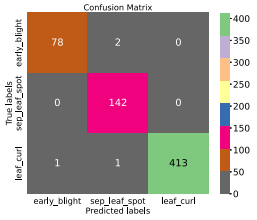
(a) Training and validation accuracy

(b) Training and validation losses

	precision	recall	f1-score	support
0	0.99	0.97	0.98	80
1	0.98	1.00	0.99	142
2	1.00	1.00	1.00	415
accuracy			0.99	637
macro avg	0.99	0.99	0.99	637
weighted avg	0.99	0.99	0.99	637

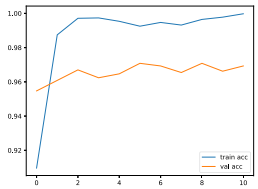
Accuracy: 0.9937

(c) Classification report

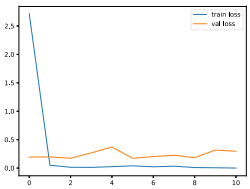


(d) Confusion matrix

Fig. 9. Gaussian Blur and the Gaussian Noise filter with RGB to HSI color conversion in ResNet-50.



(a) Training and validation accuracy

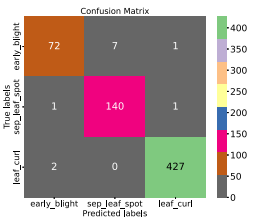


(b) Training and validation losses

	precision	recall	f1-score	support
0	0.96	0.90	0.93	80
1	0.95	0.99	0.97	142
2	1.00	1.00	1.00	429
accuracy			0.98	651
macro avg	0.97	0.96	0.96	651
weighted avg	0.98	0.98	0.98	651

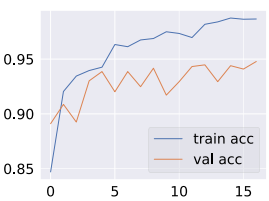
Accuracy: 0.9816

(c) Classification report

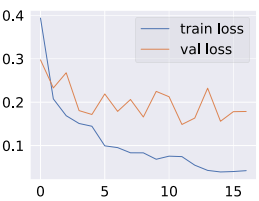


(d) Confusion matrix

Fig. 10. Median filter with RGB to CMYK color conversion in VGG-19.



(a) Training and validation accuracy

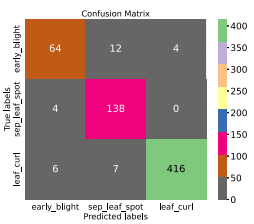


(b) Training and validation losses

	precision	recall	f1-score	support
0	0.86	0.80	0.83	80
1	0.88	0.97	0.92	142
2	0.99	0.97	0.98	429
accuracy			0.95	651
macro avg	0.91	0.91	0.91	651
weighted avg	0.95	0.95	0.95	651

Accuracy: 0.9493

(c) Classification report



(d) Confusion matrix

Fig. 11. Median filter with RGB to CMYK color conversion in MobileNet-V2.

in Fig. 9(a)). Fig. 9(b) displays the training and validation losses. The training loss in this manner is zero and the validation loss is 1.53%. For each tomato leaf disease class, classification metrics for the ResNet-50 model are given in Fig. 9(c) and Fig. 9(d) provides the confusion matrix, which depicts the actual levels and predictive levels. ResNet-50 has achieved an impressive 99.37% test accuracy in this section.

3.3. Median filter with RGB to CMYK color conversion

3.3.1. VGG-19

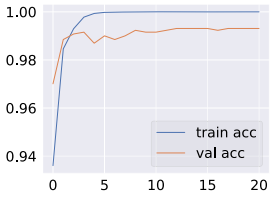
Fig. 10(a) shows the training and validation accuracy of the Median filter with RGB to CMYK color conversion in Vgg-19. The graph shows Vgg-19 has achieved 99.98% training accuracy and 96.93% validation accuracy, which is quite satisfactory. Whereas, Fig. 10(b) shows a training loss of zero and a validation loss of 29.66% for this model. In this case, the classification report is shown in Fig. 10(c) with all performance measures that are mentioned above. The confusion matrix is depicted in Fig. 10(d). The final accuracy of the Vgg-19 model is 98.16%.

3.3.2. MobileNet-V2

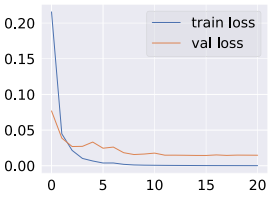
In MobileNet-V2, the training accuracy and the validation accuracy in Fig. 11(a) are 98.66% and 94.78% respectively. Whereas Fig. 11(b), which represents the training and validation losses, shows a training loss of 4.23% and a validation loss of 17.86%. Fig. 11(c) represents the classification report and Fig. 11(d) represents the Confusion Matrix of the Median filter with RGB to CMYK color conversion method applied in MobileNet-V2. The final accuracy of our MobileNet model for this method is 94.93%.

3.3.3. ResNet-50

ResNet-50 has shown 100% training accuracy and 99.31% validation accuracy, as shown in Fig. 12(a). Fig. 12(b) shows the training and validation losses. It is clear that the training loss is also zero with this method and the validation loss is 1.47%. Classification metrics for the ResNet-50 model are given in Fig. 12(c). Moreover, the confusion matrix is given in Fig. 12(d) for this methodology. Here ResNet-50 has shown a test accuracy of 98.92%.



(a) Training and validation accuracy

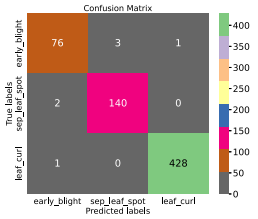


(b) Training and validation losses

	precision	recall	f1-score	support
0	0.96	0.95	0.96	80
1	0.98	0.99	0.98	142
2	1.00	1.00	1.00	429
accuracy			0.99	651
macro avg	0.98	0.98	0.98	651
weighted avg	0.99	0.99	0.99	651

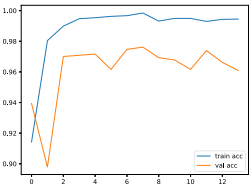
Accuracy: 0.9892

(c) Classification report

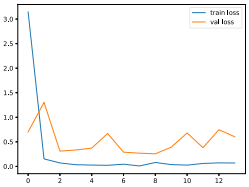


(d) Confusion matrix

Fig. 12. Median filter with RGB to CMYK color conversion in ResNet-50.



(a) Training and validation accuracy

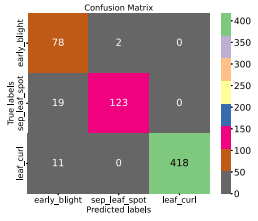


(b) Training and validation losses

	precision	recall	f1-score	support
0	0.72	0.97	0.83	80
1	0.98	0.87	0.92	142
2	1.00	0.97	0.99	429
accuracy			0.95	651
macro avg	0.90	0.94	0.91	651
weighted avg	0.96	0.95	0.95	651

Accuracy: 0.9508

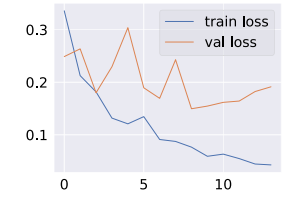
(c) Classification report



(d) Confusion matrix

Fig. 13. Median filter with RGB to HSI color conversion in VGG-19.

(a) Training and validation accuracy

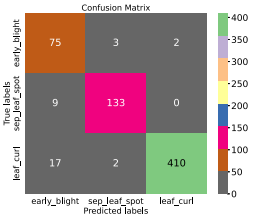


(b) Training and validation losses

	precision	recall	f1-score	support
0	0.74	0.94	0.83	80
1	0.96	0.94	0.95	142
2	1.00	0.96	0.98	429
accuracy			0.95	651
macro avg	0.90	0.94	0.92	651
weighted avg	0.96	0.95	0.95	651

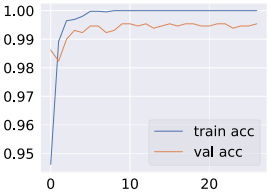
Accuracy: 0.9493

(c) Classification report

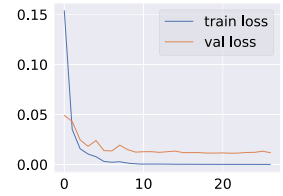


(d) Confusion matrix

Fig. 14. Median filter with RGB to HSI color conversion in MobileNet-V2.



(a) Training and validation accuracy

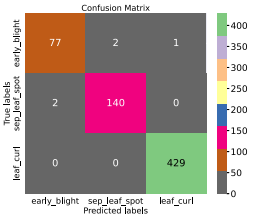


(b) Training and validation losses

	precision	recall	f1-score	support
0	0.97	0.96	0.97	80
1	0.99	0.99	0.99	142
2	1.00	1.00	1.00	429
accuracy			0.99	651
macro avg	0.99	0.98	0.98	651
weighted avg	0.99	0.99	0.99	651

Accuracy: 0.9923

(c) Classification report



(d) Confusion matrix

Fig. 15. Median filter with RGB to HSI color conversion in ResNet-50.

3.4. Median filter with RGB to HSI color conversion

3.4.1. VGG-19

The training and validation accuracy of the Median filter with RGB to HSI color conversion in VGG-19 is illustrated in Fig. 13(a). The graph clearly indicates that our Vgg-19 model achieves 99.45% training accuracy and 96.09% validation accuracy. Additionally, Fig. 13(b) displays 6.87% training loss and 60.26% validation loss. Furthermore, Fig. 13(c) presents the classification report and the corresponding confusion matrix which is presented in Fig. 13(d). In this case, our Vgg-19 model achieves a final accuracy of 95.08%.

3.4.2. MobileNet-V2

For MobileNet-V2, the training accuracy is 98.37%, and the validation accuracy is 92.48% that are given in Fig. 14(a). Now, Fig. 14(b) represents a training loss of 4.28% and a validation loss of 19.13%. Fig. 14(c) represents the classification report, and Fig. 14(d) represents the Confusion Matrix of the median filter with RGB to HSI color conversion in MobileNet-V2. The final accuracy of our MobileNet-V2 model for this method is 94.93%.

3.4.3. ResNet-50

Fig. 15(a) demonstrated 100% training accuracy and 99.54% validation accuracy. Fig. 15(b) displays that the training loss also is zero here and the validation loss is 1.18%. For each tomato leaf disease class, classification metrics for the ResNet-50 model are given in Fig. 15(c), and Fig. 15(d) provides the confusion matrix, which depicts the actual levels and predictive levels while predicting the result. ResNet-50 has achieved a satisfactory 99.23% test accuracy here.

3.5. Choosing the best model

All the results of the proposed methodologies along with the models that we have tried in this research are given in Table 1.

Table 1 clearly shows that all of our models give over 94% accuracy in every methodology. ResNet-50 performs very well as it gives over 98% accuracy for all the methods. But it gives the best results of 99.53% in Gaussian Blur and the Gaussian Noise filter with the RGB to CMYK color conversion method in the Resnet-50 model. Besides, Vgg-19 gives over 98% accuracy in every method, and MobileNet-V2 performs slightly less in comparison with our other two models in ev-

Table 1
Overall Performance Table.

Methods	Model and Accuracy
Gaussian Blur and Gaussian Noise filters with CMYK color model	Vgg19 98.27% MobileNet-V2 94.98% Resnet-50 99.53%
Gaussian Blur and Gaussian Noise filters with HSI color model	Vgg19 98.12% MobileNet-V2 95.13% Resnet-50 99.37%
Median Filter with CMYK color model	Vgg-19 98.16% MobileNet-V2 94.93% Resnet-50 98.92%
Median Filter with HSI color model	Vgg-19 95.08% MobileNet-V2 94.93% Resnet-50 99.23%

Table 2
Comparison with some of the existing works.

Reference	Proposed Model and Accuracy
Jiang et al. [9]	Resnet50 - 98%
Kibria et al. [8]	Googlenet - 99.23% VGG16 - 98%
Brahimi et al. [30]	CNN - 99.18%
Ashok, Kishore et al. [10]	CNN - 98.12%
Kumar et al. [12]	VGG16 - 99.25%, LeNet, ResNet50, and Xception (between 91% to 98.65%)
Proposed best methodology	Vgg19 98.27% MobileNet-V2 94.98% Resnet-50 99.53%

ery methodology. From Table 1, by comparing the overall performance of our selected three models in every method, we can declare the Gaussian Blur and the Gaussian Noise filters with the RGB to CMYK color conversion model as our best methodology as it provides the best result rather than the others. It shows 98.27% accuracy in Vgg-19, 94.98% in MobileNet-V2, and 99.53% in the ResNet-50 model.

3.6. Comparing our selected model with some existing works

Many works are available on tomato leaf disease detection which is done by using neural networks. Table 2 shows the performance of some existing works that have been published in recent years.
Jiang et al. [9] have used Resnet-50 as their basic model and then applied 11x11 convolution kernel to improve the result. They also used the Leaky-ReLU activation function at last to generate the best result. Finally, they gained 98% test accuracy using the Resnet-50 model. Whereas, Kibria et al. [8] have acquired 99.23% accuracy in Googlenet and 98% accuracy in VGG16 transfer learning methods. Besides, Brahimi et al. [30] applied Convolutional Neural Network in order to find out a good result in their methods. They have got 99.18% accuracy applying CNN. Another remarkable work using CNN is done by Ashok, Kishore et al. [10] who got 98.12% accuracy during the test process. Kumar et al. [12] also worked with neural networks to detect the leaf disease of tomatoes. They have applied LeNet, Rasnet-50, VGG-16, and Xception models in their research. They have got quite high accuracy in VGG-16 which is 99.25% and LeNet, ResNet50, and Xception all are between 91% to 98.65% respectively.

Now by comparing Tables 1 and 2, we can clearly say that our models are well-fitted for providing better accuracy. Also, we can conclude that the methodology which we have selected as our best provides the highest result in comparison with the other methodologies that are presented in Tables 2.

4. Conclusion

The triumph of computer and information technology continues to shine in many sectors of the world including agriculture. But most of the developing and underdeveloped countries still use ancient and underdeveloped methods for doing most of the work in the agricultural sector. Yet most people follow the manual method of diagnosing leaf disease, which is both time-dependent and somewhat complex. Once this method is applied to a physical device or in software, people will be benefited immediately regarding the detection of mentioned leaf diseases of tomatoes.
Due to the simplicity of our environmental set-up (i.e., hardware, software, etc.), we did our research on a few images. We might have gotten better results if we had trained the models with 8 to 10 times more images than this. In the future, we will try to work with a much larger dataset and also there will be various real images of various tomato leaf diseases captured from our surroundings. Moreover, we will work on building hybrid models, which means, we will try to combine decision tree, SVM, etc. machine learning models with CNN models, which may bring more accuracy and speed to our “Tomato Leaf Disease” detection system.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Funding

This research received no external funding.

References

[1] The scientific name of Tomato. The plants database, database (version 5.1.1). National Plant Data Center, NRCS, USDA. Baton Rouge, LA 70874-4490 USA, http://plants.usda.gov/Reference. (Accessed 13 March 2023), for: Solanum lycopersicum var. lycopersicum.
[2] The scientific name of Potato. The plants database, database (version 5.1.1). National Plant Data Center, NRCS, USDA. Baton Rouge, LA 70874-4490 USA, http://plants.usda.gov/Reference. (Accessed 13 March 2023), for: Solanum tuberosum.
[3] John McCarthy, Marvin L. Minsky, Nathaniel Rochester, Claude E. Shannon, A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955, AI Mag. 27 (4) (Dec. 2006) 12.
[4] Kirtan Jha, Aalap Doshi, Poojan Patel, Manan Shah, A comprehensive review on automation in agriculture using artificial intelligence, Artif. Intell. Agric. 2 (2019) 1–12.
[5] J.M. McKinion, H.E. Lemmon, Expert systems for agriculture, Comput. Electron. Agric. 1 (1) (1985) 31–40.
[6] Hal Lemmon Comax, An expert system for cotton crop management, Science 233 (4759) (1986) 29–33.
[7] Diego Inácio Patrício, Rafael Rieder, Computer vision and artificial intelligence in precision agriculture for grain crops: a systematic review, Comput. Electron. Agric. 153 (2018) 69–81.
[8] Hareem Kibriya, Rimsha Rafique, Wakeel Ahmad, S.M. Adnan, Tomato leaf disease detection using convolution neural network, in: 2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST), 2021, pp. 346–351.
[9] Ding Jiang, Fudong Li, Yuequan Yang, Song Yu, A tomato leaf diseases classification method based on deep learning, in: 2020 Chinese Control and Decision Conference (CCDC), IEEE, 2020, pp. 1446–1450.
[10] Surampalli Ashok, Gemini Kishore, Velpula Rajesh, S. Suchitra, S.G. Gino Sophia, B. Pavithra, Tomato leaf disease detection using deep learning techniques, in: 2020 5th International Conference on Communication and Electronics Systems (ICCES), 2020, pp. 979–983.
[11] Prajwala Tm, Alla Pranathi, Kandiraju SaiAshritha, Nagaratna B. Chittaragi, Shashidhar G. Koolagudi, Tomato leaf disease detection using convolutional neural networks, in: 2018 Eleventh International Conference on Contemporary Computing (IC3), 2018, pp. 1–5.

[12] Akshay Kumar, M. Vani, Image based tomato leaf disease detection, in: 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), IEEE, 2019, pp. 1–6.
[13] Emmarex, Dataset of diseased plant leaf images and corresponding labels, https://www.kaggle.com/datasets/emmarex. (Accessed 20 December 2022).
[14] David G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.
[15] Gaussian Noise. Hasty.ai, https://hasty.ai/docs/mp/augmentations/gaussian-noise. (Accessed 17 March 2023).
[16] Jahne Bernd, Digital Image Processing, a Systematic Approach to Digital Image Processing/Uses Examples and Images to Illustrate Basic Concepts, Springer, 2005.
[17] R.W.G. Hunt, Mathematical equation of HSI color model, in: Reproduction of Colour, 6 edition, John Wiley & Sons Ltd, 2004, accessed 29 March 2023.
[18] Karen Simonyan, Andrew Zisserman, Very deep convolutional networks for large-scale image recognition, preprint, arXiv:1409.1556, 2014.
[19] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen, Mobilenetv2: inverted residuals and linear bottlenecks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.
[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[21] Vinod Nair, Geoffrey E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010, pp. 807–814.
[22] John S. Bridle, Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition, Neurocomput., Found. Res. 1 (3) (1990) 145–163.
[23] Diederik P. Kingma, Jimmy Ba Adam, A method for stochastic optimization, preprint, arXiv:1412.6980, 2014.
[24] François Chollet Lin Keras, A concise api for deep learning, preprint, arXiv:1910.00547, 2019.
[25] Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016.
[26] L. Prechelt, Early Stopping–but when? 2012, pp. 53–67.
[27] David M.W. Powers, Evaluation: from precision, recall and f-factor to roc, informedness, markedness & correlation, J. Mach. Learn. Technol. 2 (1) (2011) 37–63.
[28] Jesse Davis, Mark Goadrich, The relationship between precision-recall and roc curves, in: Proceedings of the 23rd International Conference on Machine Learning, ACM, 2006, pp. 233–240.
[29] C.J. Van Rijsbergen, Information Retrieval, second edition, Butterworths, London, 1979.
[30] Mohammed Brahimi, Kamel Boukhalfa, Abdelouahab Moussaoui, Deep learning for tomato diseases: classification and symptoms visualization, J. Appl. Artif. Intell. 31 (4) (2017) 299–315.