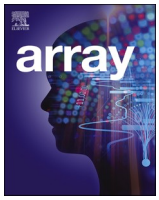
Array 19 (2023) 100313

Contents lists available at ScienceDirect

Array

journal homepage: www.sciencedirect.com/journal/array



A real-time application-based convolutional neural network approach for tomato leaf disease classification

Showmick Guha Paul [a], Al Amin Biswas [b,*], Arpa Saha [a], Md. Sabab Zulfiker [a], Nadia Afrin Ritu [c], Ifrat Zahan [a], Mushfiqur Rahman [a], Mohammad Ashraful Islam [c]

[a] Department of Computer Science and Engineering, Daffodil International University, Dhaka, Bangladesh
[b] Department of Computer Science and Engineering, Bangabandhu Sheikh Mujibur Rahman University, Kishoreganj, Bangladesh
[c] Department of Computer Science and Engineering, Jahangirnagar University, Dhaka, Bangladesh

ARTICLE INFO

Keywords:
Tomato leaf disease
Classification
Deep learning
Transfer learning
CNN
VGG-16
VGG-19

ABSTRACT

Early diagnosis and treatment of tomato leaf diseases increase a plant's production volume, efficiency, and quality. Misdiagnosis of disease by farmers can lead to an inadequate treatment strategy that hurts the tomato plants and agroecosystem. Therefore, it is crucial to detect the disease precisely. Finding a rapid, accurate approach to take care of the issue of misdiagnosis and early disease identification will be advantageous to the farmers. This study proposed a lightweight custom convolutional neural network (CNN) model and utilized transfer learning (TL)-based models VGG-16 and VGG-19 to classify tomato leaf diseases. In this study, eleven classes, one of which is healthy, are used to simulate various tomato leaf diseases. In addition, an ablation study has been performed in order to find the optimal parameters for the proposed model. Furthermore, evaluation metrics have been used to analyze and compare the performance of the proposed model with the TL-based model. The proposed model, by applying data augmentation techniques, has achieved the highest accuracy and recall of 95.00% among all the models. Finally, the best-performing model has been utilized in order to construct a Web-based and Android-based end-to-end (E2E) system for tomato cultivators to classify tomato leaf disease.

1. Introduction

1.1. Background

Identification of plant diseases is a crucial and practical issue in agriculture since disease prevention begins with an early disease diagnosis and ends with the preservation of goods. The scientific name for tomatoes is *Solanum lycopersicum*. Tomatoes are able to grow in practically any soil as long as it has adequate drainage [1]. Fig. 1 demonstrates the tomato plant cultivation in the field of Bangladesh.

One of the most extensively cultivated plants in the world is the tomato, a nutrient-dense fruit that provides farmers with a significant source of cash. In addition to being incredibly nutrient-dense, tomatoes also have pharmacological properties that protect against a number of disorders, including hypertension, hepatitis, gingival bleeding, and others [2–7]. However, several tomato plant diseases that are seen on the leaves have an impact on the amount and quality of production. As shown in Fig. 2, bacterial spot, early blight, late blight, leaf mold, septoria leaf spot, spider mites two spotted spider mite, target spot, tomato yellow leaf curl virus, tomato mosaic virus, and powdery mildew are a few well-known ailments of tomato leaves.

Early disease identification and natural treatment are essential components of effective disease management. Thus, it is crucial to understand the causes and impact of those diseases.

1.1.1. Bacterial spot

Xanthomonas vesicatoria bacteria is the cause behind bacterial spots [8]. The bacterial spot may appear in all places where tomatoes are cultivated. However, it is more usually encountered in warm, humid regions and in greenhouses. Small, water-soaked, angular circular patches that are both dark brown and black in color, with a yellow halo as an option on the leaf, are how this disease is typically identified. The potentially fatal bacterial spot disease may result in unmarketable fruit and possibly plant death in extreme circumstances.

* Corresponding author.
E-mail addresses: showmick.cse@gmail.com (S.G. Paul), alaminbiswas.cse@gmail.com (A.A. Biswas), arpasaha.cse@gmail.com (A. Saha), sabab.rumc@gmail.com (Md.S. Zulfiker), nadiaritu@juniv.edu (N.A. Ritu), ifratjahan5656@gmail.com (I. Zahan), mushfiqur.cse@diu.edu.bd (M. Rahman), ashraful.islam@juniv.edu (M.A. Islam).

https://doi.org/10.1016/j.array.2023.100313
Received 14 March 2023; Accepted 26 July 2023
Available online 3 August 2023

Fig. 1. Tomato plant cultivation in the field.

1.1.2. Early blight

Early blight is caused by the fungus Alternaria solani, which can infect tomato leaves at any time throughout the plant's growth cycle. This disease is characterized by irregular lesions near the ground, developing yellow patches that darken into concentric black rings and may have a chlorotic region surrounding the lesion. The whole foliage is destroyed by this disease, which spreads quickly to other plants [9]. Despite the fact that early blight may develop in any kind of weather, it prefers moist environments like persistent rain or even heavy dews.

1.1.3. Late blight

Late blight, caused by the fungus Phytophthora infestans, is perhaps the worst disease ever seen in tomato leaves globally, accounting for large annual economic losses. Late blight disseminates quickly because of its spore-based mode of transmission. This disease is more likely to develop in a cool, damp environment. Typically detected on newly developed leaves in the plant canopy, irregularly formed, water-soaked lesions are the earliest signs of late blight on tomato leaves. As the disease worsens, lesions become larger, and the affected leaves turn brown, shrivel, and die [9].

1.1.4. Leaf mold

The fungus Passalora fulva is responsible for the development of tomato leaf mold. The top surfaces of leaves develop small, round, pale greenish-yellow patches with blurry, undefinable edges. Clusters of brown dots are usually developed on leaves [10]. The fungus settles on leaves and penetrates the plant's stomata, which are utilized for gas exchange.

1.1.5. Septoria leaf spot

The fungus responsible for septoria leaf spots is Septoria lycopersici. The fungus is transmitted from the soil to the plants by splashing water, insects, and tools. Septoria leaf spots first appear as round spots surrounded by a yellow-haloed dot on the undersides of older leaves. The patches will grow and maybe blend together as the disease progresses. Septoria leaf spot may not be deadly for tomato plants, but it may swiftly weaken and defoliate them, preventing plants from producing mature fruit and reducing the fruits' size and quality [11].

1.1.6. Spider mites two spotted spider mite

Tetranychus urticae, often known as the two spotted spider mite, is a common polyphagous arthropod pest that has a significant negative economic impact on the tomato crop. Affected leaves develop a brown or yellow coating on their undersides. The whole plant may be covered in fine webbing when the two spotted spider mite is heavily infested [12].

1.1.7. Target spot

Corynespora cassiicola is a fungus that causes tomato target spot. Regions of the world that are warm year-round experience the disease on field-grown tomato leaves. Initially, the leaves will develop tiny, water-filled patches. The spots develop into small, necrotic lesions that have light brown centers and dark margins [8]. Target spot infections lower production in two ways: indirectly by lowering the photosynthetic area and directly by making the fruit less marketable due to fruit spots.

1.1.8. Tomato yellow leaf curl virus

Despite not being seed-borne, whiteflies are able to spread the virus named Tomato yellow leaf curl. The affected plant has several symptoms, such as leaf edges that roll upward and inward, yellowing leaflets, smaller leaves than usual, dropping off flowers, and elevated scabby patches [13]. Furthermore, in disease-affected plants, no fruit development might happen.

1.1.9. Tomato mosaic virus

The leaves dropped early because of the Tomato mosaic virus. The production of distorted leaflets, small leaves, yellowing leaves, and systemic necrotic patterns. Furthermore, the quality and growth of fruit are affected by the virus [14].

1.1.10. Healthy

Healthy leaves have vigor, a uniform color (unless variegated), open growth, and an upright look.

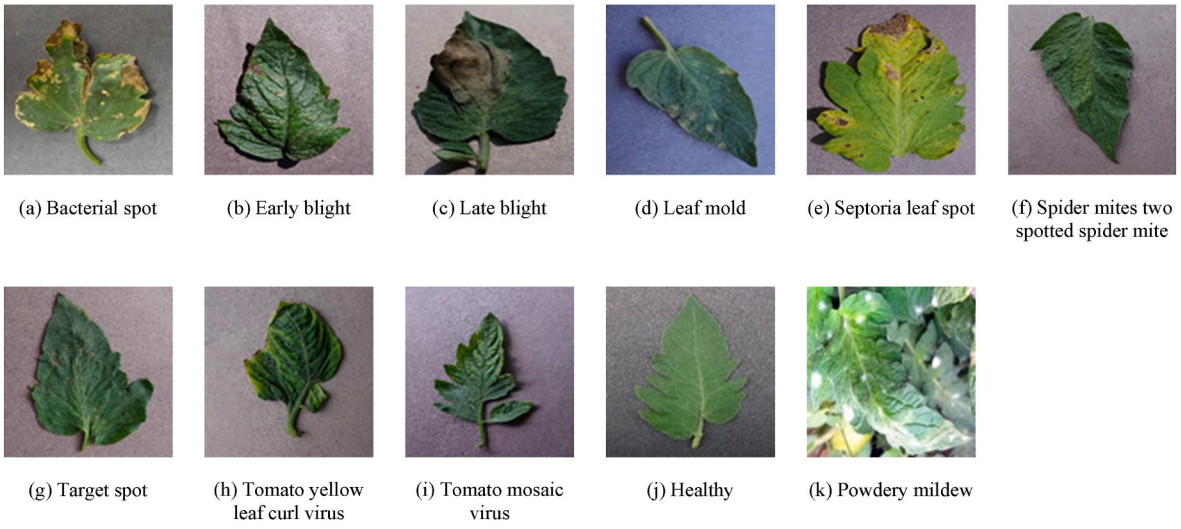


Fig. 2. Sample images (random) for each class of tomato leaf.

1.1.11. Powdery mildew

The obligatory parasite Oidium neolycopersici causes powdery mildew on tomato leaves and obtains nutrients. Mycelium produced by the fungus grows on unpaid host plants that are either active or dormant. It's more frequent in commercial tomato fields and greenhouses than at home [15]. It hurts plants and diminishes output wherever it flourishes.

The precise and timely classification of tomato leaf diseases is of the utmost importance for disease management. It is widely accepted that diseases are one of the most significant reasons for decreased productivity. In addition, disease caused significant losses to the agricultural economy associated with tomato farming. For instance, the most prevalent diseases that may severely impact crop output are early blight and late blight [16].

1.2. Motivation

Plants are essential to life as they provide us with food, fiber, shelter, medicine, and fuel. Plants encounter several challenges despite being essential to life. Without accurate identification of tomato leaf diseases, the quality and quantity of tomato production can decrease. This further harm a nation's economy [17]. The United Nations Food and Agriculture Organization (FAO) estimates that in order to fulfill future food demands, agricultural production must increase by 70% by 2050 [18]. The agricultural ecology is significantly impacted by disease-prevention agents like fungicides and bactericides. Farmers may incorrectly identify a disease, leading to a less optimal treatment approach that ultimately harms the plant. Furthermore, field trips for domain experts are expensive and require time. Therefore, the agroecosystem requires a rapid and efficient disease classification system. Improvements in disease detection technologies, such as image processing and a CNN-based model, will allow for the creation of systems capable of early disease classification on tomato leaves. The danger of production loss, processing costs, and the negative environmental consequences of chemical inputs (contamination with soil and water) can be mitigated when diseases can be detected in their early stages and the appropriate procedures are taken to stop the disease's spread [19].

1.3. Contribution

I. Various transfer-learning-based CNN models and a custom CNN model are utilized for classifying tomato leaf disease of ten classes with one healthy class.
II. This study analyzed the relationship among performance, data augmentation, and ablation studies.
III. This study proposes a custom lightweight and efficient CNN that has achieved adequate classification accuracy and outperformed the majority of earlier studies on tomato leaf disease.
IV. The final model has been deployed in Web-based and Android-based applications to aid tomato cultivators in disease classification.

Following is a breakdown of the remaining parts of this research work: the related work on the subject of disease classification is discussed in Section 2. Section 3 describes the TL-based models and proposed custom CNN architectures for tomato leaf disease classification, as well as the training procedure and parameters. Section 4 describes the tomato leaf disease dataset as well as its preprocessing for training purpose. Additionally, the evaluation metrics utilized in the study are described. Furthermore, the experimental results, performance analysis, comparison of results, and E2E system deployment procedure are also included in this section. In Section 5, the study is concluded and the potential future research directions are highlighted.

2. Related work

The production quantity and quality of tomato plants are affected by the early identification of tomato plant disease. Various studies have been performed related to tomato disease classification using diverse techniques and procedures. The use of deep learning (DL) techniques significantly increases the classification accuracy of image-related tasks. Researchers adopted machine learning and deep-learning techniques for tomato leaf disease classification are reviewed here.

In order to classify tomato leaf disease, TM et al. [20] added an extra block to the LeNet architecture, which consisted of convolution layers, activation layers, and pooling layers. For reducing the time needed for the training process, the dataset images have been resized by 60 × 60 pixel size and has achieved an accuracy of 94.00–95.00%. Despite its effectiveness, nine tomato leaf diseases and one healthy tomato leaf class have been considered in the study.

Deep learning models based on transfer learning are used for disease classification as they can achieve satisfactory results. Therefore, to build a custom CNN-based model, it's required to compare it with the transfer-learning models to justify the improvement. Agarwal et al. [21] developed a simplified and effective CNN model with three convolutions, three max-pooling, and two fully connected layers. By applying data augmentation techniques, new sample images are generated to balance image quantity across all ten classes. With the comparison of the proposed model with the VGG-16, MobileNet, and InceptionV3, the proposed model has achieved an improved accuracy of 91.20%.

Despite data augmentation, other factors such as batch size and the optimizer may affect the model's performance. Zhang et al. [22] conducted a study that included eight diseases class and one healthy tomato leaf class, and analyzed the rule of batch size and used optimizers on the performance of transfer-learning models by applying various combination. Among different optimization techniques used on various deep learning models, ResNet with the stochastic gradient descent (SGD) optimizer has achieved the highest accuracy of 96.51%, while other models' performances have been significantly increased by the use of SGD.

The dataset acquisition procedure has an impact on the effectiveness of the applied models. Ahmad et al. [23] have used two tomato leaf disease image datasets, one laboratory-based and the other a dataset that has been self-collected from the field. With the use of feature extraction and parameter tuning on ResNet, VGG-19, VGG-16, and InceptionV3 models, the performance of various metrics shows a variance in the range of 10.00%–15.00%. Among the variances, InceptionV3 using parameter tuning has obtained the highest accuracy value of 99.60% based on the laboratory-based dataset.

Although DL models are widely used to classify diseases of tomato leaves. Some researchers have extracted features from the image dataset to perform classification using ML algorithms. Basavaiah et al. [19] have used a dataset containing five classes of tomato leaf and extracted features such as hu moments, color histograms, local binary patterns, and haralick to perform classification. Various ML algorithms have been used for classification, among those methods, RF has achieved a higher accuracy of 94.00%. The proposed mechanism, however, is not automatic, and less number of the tomato leaf disease has been covered.

Various CNN-based architectures have been used not only for tomato leaf disease classification but also for other plant-related disease classifications. Mia et al. [24] have utilized CNN-based TL and traditional ML techniques and compared their performance for the classification of cucumber disease. In the study, a total of 525 image samples representing six disease classes have been acquired and preprocessed. The sample size has been increased by applying data augmentation techniques, with a total sample size of 4200 images. In the next phase, various ML algorithms has been applied. Furthermore, transfer-learning-based VGG-16, InceptionV3, and MobileNetV2 have been applied, and a comparison has been performed. Among all the procedures, CNN-based MobileNet has achieved the highest accuracy of 93.23%.

Biswas et al. [25] have used several ML classifiers based on segmentation-based feature extraction and ranking approaches to

classify carrot disease. The study employed a dataset consisting of two categories of carrots, totaling 599 images. Using the histogram equalization approach, which redistributes the intensity values in the image, the contrast of the carrot images has been boosted. Using K-means clustering, further segmentation of the carrot disease has been performed, and the top ten features have been extracted. In order to address the imbalanced dataset, Synthetic Minority Oversampling Technique (SMOTE) approaches have increased the minority class sample size. The resulting processed dataset has been then classified using a variety of ML algorithms, including Random Forest (RF), Adaptive Boosting (AB), Decision Tree (DT), Gradient Boosting (GB), and Bagging classifier (BAG). Among various techniques, RF has obtained the highest accuracy of 94.17% by utilizing the top nine ranking features.

In another study, the disease classification of tomato leaves has been conducted by Chen et al. [26], utilizing the combination of B-ARNet and ABCK-BWTR. Images consisting of five classes are denoised and enhanced by the Binary Wavelet Transform combined with Retinex (BWTR). Furthermore, background separation of images has been performed using KSW optimization by the Artificial Bee Colony Algorithm (ABCK) and classified using the Both-channel Residual Attention Network (B-ARNet), achieving an accuracy of 89.00%.

Thangaraj et al. [27] used a TL-based DNN model to recognize nine classes of tomato leaf disease and one class of healthy leaf. Different optimizers such as SGD, Adam, and RMSprop have been used to evaluate their effects on the performance of the TL model. Among the used optimizers, with the application of an Adam optimizer, the Modifed-Xception model has been able to achieve a better accuracy of 99.55%.

Tomato leaf disease has been classified using DCNN, which includes residual blocks and attention extraction modules performed by Zhao et al. [28]. The dataset contains ten classes, including one healthy class, with a sample size of 4585, which has been further augmented with the count of 22925 images. Among the ResNet-50 and SE-ResNet-50 models, SE-ResNet-50 has achieved the highest accuracy of 96.81%.

In another study by Kannan E et al. [29], a pre-trained ResNet model has been used to classify tomato leaf disease. Six classes of leaves are used in the study, with a sample size of 12,206 images. Further augmentation has been applied to the dataset to increase the image count. The author achieves an accuracy of 97.00% by applying the ResNet-50.

One of the contributing factors of research work is the deployment of the trained model on a Web or other application-based platforms, which provides an E2E solution for farmers to identify suspected disease and take early measurements. Elhassouny and Smarandache [30] have built a CNN-based model, which has been deployed on a mobile application to recognize tomato leaf disease. For building a model based on CNN, the MobileNet model has been applied to the nine most common diseases, with one healthy class. A total sample size of 7176 images has been used for training the model, which has obtained the highest accuracy of 90.30%.

Chen et al. [31] applied AlexNet to classify tomato leaf diseases and implemented the model on the Android platform. For that purpose, a dataset with 18345 training samples and 4585 testing samples from ten classes, including one healthy class, has been utilized. By using the Adam optimizer, the highest accuracy of 98.00% has been achieved. Furthermore, the trained model has been deployed in a mobile application that can be used to identify tomato leaf disease.

H. S. and Sarojadevi H [32]. used devised mechanisms such as CNN, fuzzy support vector machine (fuzzy-SVM), and region-based CNN (R–CNN) in another study. Six diseases and one class of healthy tomato leaves, containing a total of 735 image samples, have been utilized in the study. Among the techniques, the R–CNN–based model has achieved the highest accuracy of 96.73%.

According to the previous discussions, it can be concluded that deep CNN based techniques have not yet been used to classify the eleven classes of tomato leaf containing ten classes of disease. This study paves the path for future efforts to address this gap. This study utilized different transfer-learning-based CNN models and proposed a custom CNN model to classify tomato leaf disease. Furthermore, augmentation techniques have been applied and ablation studies have been performed to select the optimal parameter for training the models and to increase the models' performance. Finally, the best performing model has been deployed using the Web-based and Android-application-based platforms, which can help the tomato cultivators in classifying the tomato leaf disease.

3. Proposed system architecture and methodology

3.1. System architecture

Fig. 3 illustrates a systematic representation of our proposed system architecture. The system consists of two phases: the building phase, during which the models are trained, and the deployment phase, during which the development of E2E systems and the prediction of tomato leaf disease are performed. During the building phase, data has been collected, preprocessed, and trained with multiple models. To predict tomato leaf disease, a custom CNN model has been built from scratch. The ImageNet dataset pretrained-weight has been utilized to train the TL-based VGG-16 and VGG-19 models. The last phase is the deployment phase, in which the best-performing models are chosen according to the evaluation criteria and then implemented in the local Web-based system and Android application to predict tomato leaf disease.

3.2. Methodology

The proposed approach includes different steps. Fig. 4 illustrates a systematic representation of implementation step. In the first step, the dataset containing eleven classes, including one healthy leaf class, has been acquired. This dataset has been preprocessed by resizing the images, converting them to jpg format, and applying data augmentation techniques. The data augmentation mechanism helps to multiply the sample size of training data as well as increase the complexity of the dataset, which makes the DL model more robust and helps the model to generalize common patterns. In the next phase, a custom-built lightweight, and efficient CNN has been applied, and various other CNN-based TL models such as VGG-16, VGG-19 have been applied to the dataset. To extract features and create feature tensors, pre-trained ImageNet weights have been used for TL-based CNN models. In the implementation procedure, first, a preprocessed dataset without augmentation has been applied to the Proposed, VGG-16 and VGG-19 models, and their performance has been evaluated. In another phase, models have been applied to the augmented dataset, and their performance has been evaluated. The overall performance of all the models with and without augmentation techniques applied has been analyzed and compared. To select the optimal parameters of the proposed custom CNN model and data augmentation techniques, an ablation study has been performed. Ablation study refers to a process, where a certain part of the DL model's architecture is removed in order to understand the model's behavior. To achieve better performance, it is important to analyze the model's behavior and select the most optimal parameter. By removing certain parts, the influence of the removed part on the model and on the performance can be observed, which helps to identify and select the most effective parameters. By performing the ablation study, the proposed custom CNN model architecture and its most optimal parameters for the construction of the model have been identified. Furthermore, an optimal set of data augmentation techniques that aid the model in obtaining better performance has been considered. Additionally, a callbacks function is used during the training process to select the best-performing models based on the validation loss score. The callbacks function assists in gaining control of the entire training process by comparing and selecting the best models after each successful epoch. As the validation loss has been set as the callbacks function's selecting

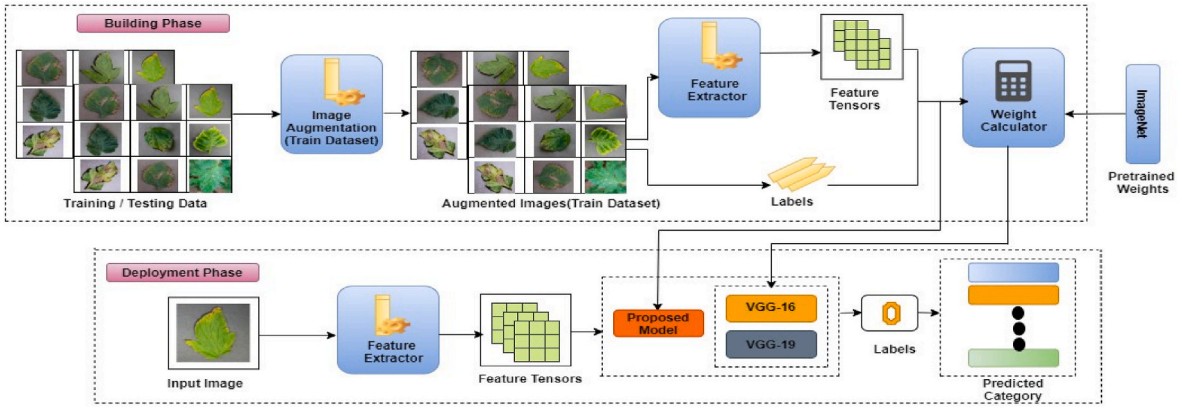


Fig. 3. Proposed system architecture (A systematic representation of our proposed approach).

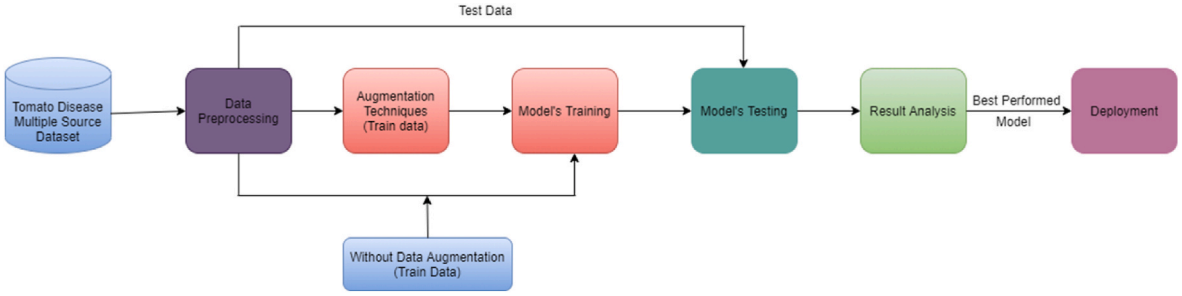


Fig. 4. Diagram of the implementation step for classifying tomato leaf disease.

criteria, the model with the least amount of validation loss is usually saved. In the next phase, the performance of the models with and without augmentation with minimal validation loss has been evaluated and compared. Finally, the best performing model has been deployed in local web-based system and an Android based smartphone application to predict disease in tomato leaves.

3.3. Basic CNN

CNN is a type of neural network, which allows the extraction of a huge quantity of features from images and use those features for better classification. CNN architectures take the raw image as input and automatically extract numerous features and, based on those features, produce output. To produce output, various parts of the CNN architecture, commonly known as the building blocks of CNN, are utilized in various combinations. There are three types of layers commonly utilized to build CNN, such as the convolution layer, the pooling layer, and the fully connected layer. Apart from those layers, other layers such as dropout, batch normalization, etc. are also often utilized. In the conventional layer, a feature map is generated from the input image and passed to the next layer to learn various patterns from the feature map. Most studies then used pooling layers, whose primary task is to reduce the feature map size in order to reduce computing costs. The pooling layer's basic principle is to summarize the features generated by the convolution layers. The activation function is applied right after a convolution layer and then that output is max pooled. The activation function or transfer function determines how a layer's nodes convert the weighted sum of input into an output. Finally, after flattening the output of the previous layer, the process of categorization begins at the fully connected layer. In the last layer, Sigmoid or SoftMax activation functions are used to produce output.

3.4. Proposed custom CNN

The proposed custom CNN utilized the preprocessed image as input to train from scratch to classify diseases of tomato leaves. The 224 × 224 image dimensions with three color channels are utilized as input by the proposed model. This model consists of the following set of layers: 4 convolutions, 4 max pooling, 3 dropouts, one FC layer, and finally dense layer with SoftMax activation function (see Fig. 5). The custom model's layers and parameters have been selected, by prioritizing accuracy and computational complexity through an ablation study. During an ablation study, specific elements of the network are eliminated to acquire a better knowledge of the network's behavior and to determine the final layers for the custom model. Table 1 presents the parameters that have been selected after performing the ablation study. These parameters have been used to construct the proposed model.

The basic principles and working concepts of the convolution layers, polling layer, FC, and activation function are discussed in the previous

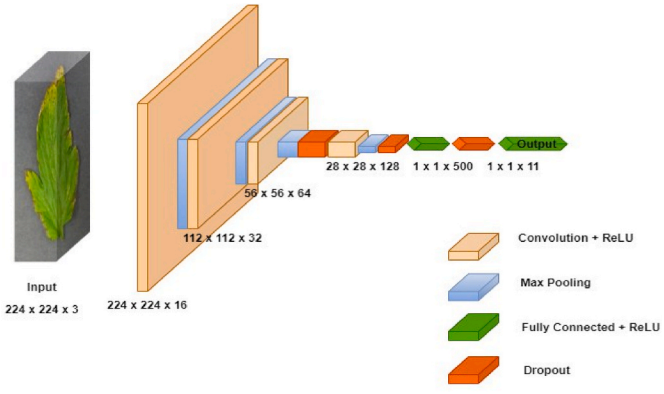


Fig. 5. Proposed custom CNN architecture.

Table 1
Training parameter for the proposed custom CNN model.

Parameter	Description
Optimization algorithm	Adam optimizer
Learning rate (α)	0.001
Weight initialization	Glorot uniform
Batch size	32
Number of epochs	100
Dropout rate	0.2 or 20%
Loss function	Categorical cross entropy
Activation function (Hidden layers)	Relu
Activation function (Output layer)	SoftMax

subsection.

To initialize the weight of the proposed custom CNN architecture, Xavier Glorot uniform weight initialization has been implemented. When performing Glorot, the weight (w) for the neuron present in hidden layers is chosen at random from a uniform distribution. The representation of Glorot weight initialization is represented by Eq. (1). The values of the weights fall somewhere between + r and -r in a random uniform distribution [33]. Where,

$$r = \sqrt{\frac{6}{X_i + X_o}} \tag{1}$$

here, in Eq. (1), X_i represents the input, and X_o represents the output connection. When the number of input and output connections are equal, the weight will be as shown in Eq. (2),

$$r = \sqrt{\frac{3}{X_i}} \tag{2}$$

To avoid linearity, activation functions must be used. Without them, the data in the network would be passed across the nodes and layers using just linear functions. The composition of these linear functions is another linear function, so regardless of how many layers the data passes through, the output is always a linear function.

The activation function in the CNN architecture determines how a layer's nodes convert the weighted sum of input into an output [34].

$$Y = Activation\ function\left(\sum (W1 \times X1 + W2 \times X2 + \ldots + W_n \times X_n + bias)\right) \tag{3}$$

The representation of activation function is represented by Eq. (3). Using Eq. (3), after the multiplication of inputs and weights the result has been added and an additional bias is applied, the activation function is applied. Within the proposed CNN model, the Relu activation function has been implemented. The Relu activation function is defined mathematically as follows Eq. (4):

$$y = max(0, x) \tag{4}$$

$$f(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases} \tag{5}$$

Here, by applying Eq. (5), the Relu activation function's output is calculated as either 0 or x. This means that there are no flat areas where saturation can happen, unless x is negative, which makes gradient descent happen very quickly. It is the most popular choice of activation function for hidden layers since it is extremely easy to compute.

Furthermore, the Adam optimizer has been employed to find the optimal node weight and minimize prediction error. Adam is derived from adaptive moment estimation. The Adam optimizer modifies the learning rate for each network weight independently, while SGD utilizes a single learning rate throughout the training process. With its many advantages, the Adam optimizer is highly recommended as a default optimization technique and serves as a useful benchmark for deep learning projects. Compared to other optimization algorithms, this one runs more quickly, uses less memory, and needs less tuning. The Adam optimizer is a combination of gradient descent with momentum and the RMSP algorithm. Adam takes over the attributes of the two optimizers and builds upon them to give a more optimized gradient descent.

$$m_t = \beta_1 * m_t + (1 - \beta_1) * \left(\frac{\delta L}{\delta \omega_t}\right)$$

$$v_t = \beta_2 * v_t + (1 - \beta_2) * \left(\frac{\delta L}{\delta \omega_t}\right)^2$$

$$\widehat{m}_t = \frac{m_t}{\left(1 - \beta_1^t\right)}$$

$$\widehat{v}_t = \frac{v_t}{\left(1 - \beta_2^t\right)}$$

$$\omega_t = \omega(t-1) - \alpha * \left(\frac{\widehat{m}_t}{\sqrt{(\widehat{v}_t)} + e}\right) \tag{6}$$

Here Eq. (6) presents the formula for Adam's optimizer. In the above equations, m_t is aggregate of gradients at time t. On the other hand, v_t represents the attribute of Root Mean Square Propagation (RMSP) algorithms. Initially, m_t and v_t are set to 0 and tend to be more biased [35]. Therefore, Adam optimizer solved the biased problem by calculated bias-corrected $\widehat{m}_t$ and $\widehat{v}_t$. Finally, by integrating the formula new weight ω_t are computed.

All of the convolution layers employed 2 × 2 kernel size with padding kept as "same" and used the Relu activation function. The Relu is a non-linear activation function whose main advantage is that it does not activate all the neurons at the same time. The kernel size of 2 × 2 is used in max pooling layers. Furthermore, in the dropout layers, dropout is set to 0.2. The 0.2 value of dropout means approximately 20.00% of the neurons randomly deactivate at a time, which helps reduce the overfitting problem of the model.

In the output layer, the SoftMax activation function is used, which is used for multiclass classification. The output of a SoftMax is a vector containing probabilities for each class. The vector's probability for all potential outcomes or classes sums to one.

Categorical Cross-Entropy loss has been used in the study. Cross-Entropy loss, often known as log loss, is a performance metric for classification models whose output is a probability value between 0 and 1. As the predicted probability gets closer 1, log loss drops gradually. Cross-Entropy loss grows when the predicted probability deviates from the actual label, and as a result, errors are penalized. Categorical Cross-Entropy loss also called SoftMax loss is combination of SoftMax activation function and Cross-Entropy loss [36]. Equation (7) defines the Categorical Cross-Entropy loss formula.

$$CE = -\sum_{i}^{C} t_i \log\left(f(s)_i\right) \tag{7}$$

where, in Eq. (8), for each class i in C, t_i and s_i represent the ground truth and the CNN score. Before calculating the Categorical CE loss, a SoftMax activation function is applied to the score, where $f(s)_i$ represents the activations. The formula of the SoftMax activation function is presented in Eq. (8).

$$\mathrm{f(s)_i} = \frac{\mathrm{e^{s_i}}}{\sum_{\mathrm{j}}^{\mathrm{C}} \mathrm{e^{s_j}}} \tag{8}$$

The model has been run for 100 epochs, while the batch size is 32. The batch size of 32 indicates that images are batched with 32 quantity to train the models at a time. By utilizing the batch size, the model's training time can be shortened. The batch size employed in the model has been determined by the ablation study. By performing the ablation

study on batch size, it has been observed that there is a strong association between the batch size and model performance. Using minimum validation loss as the selection criterion, the best model has been stored as the final model using the callbacks function.

3.5. Transfer learning

Transfer-learning is a technique where a pre-trained model is used as a base for a new or different study. This technique is very popular in DL and is used for image classification task. Developing a CNN model from scratch demands a huge dataset and a lot of time. To reduce the complexity, transfer-learning techniques are used, which reduce the amount of time and can train on limited quantities of data samples. Numerous research institutes publish models trained on vast and complex datasets that have been open-sourced, and those models can be used for other research work. The pre-trained model can then be used to build a model for the different tasks. All or a portion of the model may be used, depending on the modeling approach chosen. Those pre-trained models are usually trained on a large and challenging image classification dataset such as ImageNet dataset, which contains 1000 image categories. There are many pre-trained models that are available online and can be used for developing models [37]. Fig. 6 represents the workflow of transfer learning for the new study. Every models' various parameters, such as time, and size vary depending on the models' architecture. Therefore, it is very important to select the effective models for better performance.

3.5.1. VGG-16

The Visual Geometry Group (VGG) is one kind of CNN, which is now among the top computer vision models available. The 16 refers to the 16 weighted layers present in the networks. The VGG-16 algorithm can recognize 1000 different classes of objects and classify them accordingly. Fig. 7 illustrates a representation of VGG-16 architecture.VGG-16 takes the input of an image pixel size of 224 × 224 with 3 RGB channels. There is a total of 21 layers in VGG-16 model: 13 convolutional, 5 max pooling, and 3 dense layers. Here, only 16 are learnable parameter layers. In the convolution layers, with padding as same, a 3 × 3 size filter with stride 1 is used. A filter with a stride of two and dimensions of 2 × 2 is used for the max pooling layers. The architecture shows that convolution and max pool layers are consistently arranged throughout the architecture [38]. Furthermore, all the extracted features have been then flattened. Finally, the model is created by inheriting the convolution layers of VGG-16 and adding two new fully connected layers with configurable number of neurons (128,64) in first two layers and 11 neurons in last layer with SoftMax activation function. Here, fully connected layers are retrained fully.

3.5.2. VGG-19

VGG-19 is the same as VGG-16 except that it contains 19 learnable parameter layers. Fig. 8 illustrates a representation of VGG-19 architecture. In the input, layer images are used with a pixel size of 224 × 224 and 3 RGB channels. The same filter sizes used in VGG-16 are used in the convolution layers of VGG-19. Furthermore, VGG-19 used the same size of the max pooling layer as VGG-16 [39]. The main difference between VGG-16 and VGG-19 is that from the 3rd to 5th blocks, an additional convolution layer is used. Moreover, the model is created by inheriting the convolution layers of VGG-19 and adding two new fully connected layers with configurable number of neurons (128,64) in first two layers and 11 neurons in last layer with SoftMax activation function. Here, fully connected layers are retrained fully.

4. Experimental evaluation findings and analysis

4.1. Environment specification

Table 2 represents the experimental setup of the study. The research work takes place on an Intel i7-10510U CPU with 1.80 GHz processing power. Furthermore, the machine has 24 GB of RAM and an integrated graphics card. To train the models, an NVIDIA GeForce MX250 graphics card with 4 GB of dedicated GPU memory has been used. All of the models are built in Python and run on deep learning libraries like Keras and TensorFlow.

4.2. Dataset

The study is performed by using the "tomato disease multiple sources" named dataset, which contains images of tomato leaf disease [40]. The image in the dataset is captured from both laboratory and in-the-wild scenes. Furthermore, this dataset of images has been collected from various sources.

Table 3 represent the dataset used in the study. In the dataset, tomato leaves have a total of 32535 images and are classified into eleven classes. Among the classes, one contains sample images of healthy tomato leaves, and the rest of the classes contain various disease-affected tomato leaf image samples. Additionally, the dataset consists of images in several image formats, such as jpg and png. The dataset was pre-split into 2 sections: train data, which contains approximately 80% of the total images, and valid data (considered as test data), which contains approximately 20% of the total images.

4.3. Preprocessing

First, all of the images are converted to jpg format during the preprocessing stages. Images in the dataset are scaled to a resolution of 224 × 224 pixels to facilitate a faster training time and make model training computationally feasible. The validation dataset has been created using 10.00% of the training dataset. The final training dataset for this study has been created using data augmentation techniques. Data augmentation refers to a set of techniques for generating fictional data points from pre-existing data. Data augmentation enriches and complicates data,

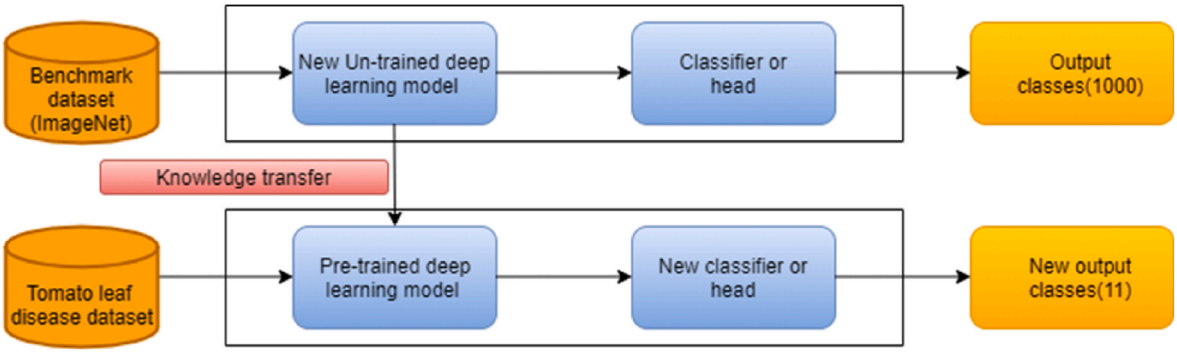


Fig. 6. Transfer learning procedure.

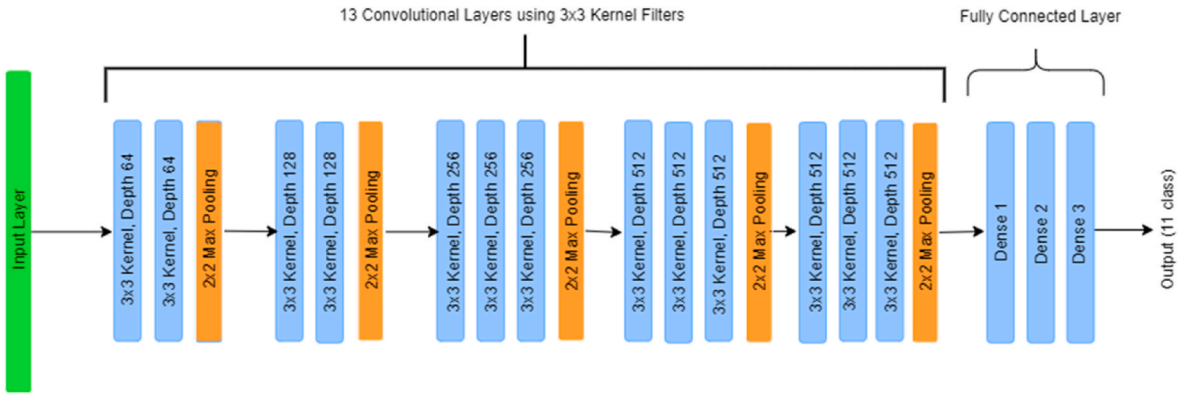


Fig. 7. VGG-16 architecture.

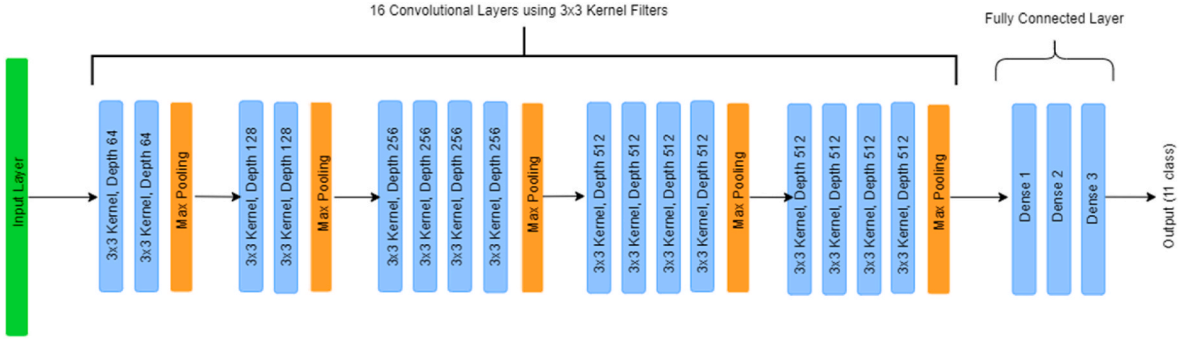


Fig. 8. VGG-19 architecture.

Table 2
Experimental setup.

Process name	S. N.	Action
Input	1.	Collected images of 11 classes of tomato leaf, including 10 classes of disease.
Environment	2.	Anaconda, Jupyter Notebook.
Configuration	3.	Import all necessary libraries and packages.
	4.	Load the images.
Directories Configuration	5.	Load directories for training, testing, and create validation (10% of training data).
Training and Testing	6.	Build custom CNN models. For transfer learning, use
	7.	models trained on the ImageNet dataset. Fine-tune the models by adding the fully connected layer and the SoftMax activation function.
Model Compilation	8.	The model compiles with an Adam optimizer and a
	9.	learning rate of 0.001.
	10.	Set 100 epochs for model training.
	11.	As a model checkpoint, use the validation loss to monitor. Save model.
Performance Evaluation	12.	Generate classification report.
	13.	Generate models' accuracy and loss reports.
	14.	Generate ROC-AUC Curve.
Prediction	15.	Load best model.
	16.	Load random images.
	17.	Predict disease classes.

improving model performance and accuracy. Data augmentation techniques reduce operational costs by introducing transformations into the datasets. To get the higher accuracy model, complex data is essential, and data augmentation helps with that. Table 4 describes the hyperparameter and its corresponding value employed in this study. Fig. 9 illustrates a random sample image and its various augmented forms.

4.4. Evaluation metrics

The effectiveness of the statistical, ML, or DL model is assessed using evaluation metrics. To evaluate a study's proposed model, it is crucial to use a variety of evaluation metrics. Metrics for evaluation are crucial for ensuring models' performance. Model predictive or classification efficacy can be measured by accuracy, precision, recall, and the f1-score [41].

4.4.1. Accuracy

Accuracy is the percentage of images correctly predicted from all the predictions. The following Eq. (9) describes how the accuracy is stated:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

The accuracy score measures the number of correct predictions (TP + TN) made by a model in relation to the total number of predictions (TP + TN + FP + FN) made. The abbreviations for "true positive", "true negative", "false positive", and "false negative" are "TP", "TN", "FP", and "FN" respectively.

4.4.2. Precision

The precision, which measures the percentage of truly right positive outcomes, is determined by Eq. (10):

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

4.4.3. Recall

By comparing the number of true positive findings to the total number of actual positive samples (TP + FN), the recall value is used to gauge the accuracy of positive predictions. The following Eq. (11) is used to determine the recall value:

Table 3
The tomato leaf disease dataset.

Class ID	Class Name	Count	Training	Testing	Leaf Symptoms
C0	Bacterial spot	3558	2826	732	Small, water-soaked, angular circular patches dark brown and black in color.
C1	Early blight	3098	2455	643	Black or brown spots appear, leaf spots often have yellow or green concentric ring pattern.
C2	Late blight	3905	3113	792	Water-soaked area appears and rapidly enlarges to form purple-brown, oily-appearing blotches.
C3	Leaf mold	3493	2754	739	Irregular yellow or green area appears.
C4	Septoria leaf spot	3628	2882	746	Round spots, marginal brown, chlorotic yellow, appear.
C5	Spider mites two spotted spider mite	2182	1747	435	Show white or yellow spots, blade back netting.
C6	Target spot	2284	1827	457	Pinpoint-sized, water-soaked spots on the upper leaf surface.
C7	Tomato yellow leaf curl virus	2537	2039	498	Develop small and curl upward, crumpling, and marginal yellowing, bushy appearance.
C8	Tomato mosaic virus	2737	2153	584	Leaflets, small leaves, yellowing leaves, and systemic necrotic patterns.
C9	Healthy	3857	3051	806	Softly fuzzed, medium-green leaves.
C10	Powdery mildew	1256	1004	252	Light green and yellow blotches on leaves.
		Total: 32535	Total: 25851	Total: 6684	

The train dataset contains 25851 sample images, and 6684 images are present in the test dataset. Furthermore, 10.00% of the training dataset is used to produce the validation dataset.

Table 4
Hyperparameter for image data augmentation.

Techniques	Parameter
Shear range	0.2 or 20.00%
Zoom range	0.2 or Zoom in and Zoom out by 20.00%
Horizontal flip	True
Vertical flip	True
Width shift range	0.1 or 10.00%
Height shift range	0.1 or 10.00%

$$Recall\ or\ TPR = \frac{TP}{TP + FN} \tag{11}$$

4.4.4. F1-Score

To measure how well a model performs, researchers utilize measures like the F1-score, which is calculated by taking the harmonic mean of the model's precision and recall. It is defined as Eq. (12):

$$F1 - Score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{12}$$

4.5. Result analysis and deployment

4.5.1. Result analysis

This section describes the performance of various models that have been utilized.

Accuracy and loss graphs are used to better understand the model's behavior. The ROC-AUC curve of all the applied models has been demonstrated. Furthermore, the proposed model has been compared to other studies to determine its relevance.

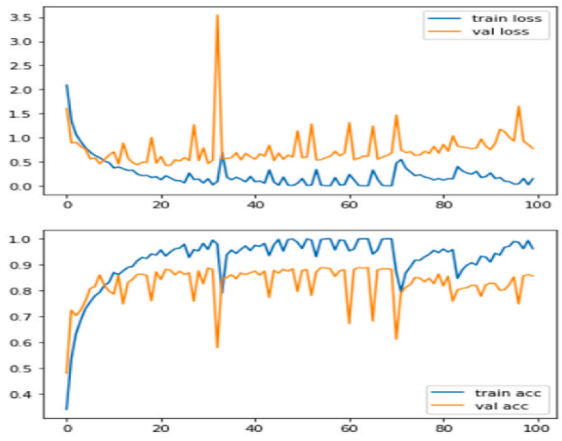
A learning graph is a graph that depicts time or epoch on one axis and learning on the other axis. Learning curves are widely used in Deep CNN (DCNN). The generalization ability of the model may be estimated by the evaluation of the validation dataset. During the training process, dual learning curves has been generated for the DL model by applying it to both the training dataset and the validation dataset. There are two types of learning curves:

The accuracy graph indicates the obtained accuracy of training and validation after each epoch run. The loss graph, on the other hand, shows how much loss training and validation experienced.

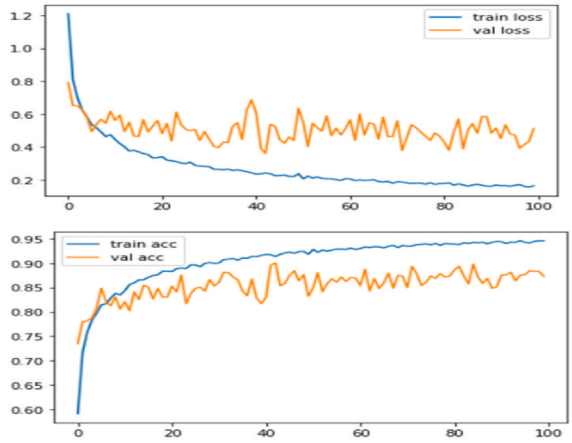
By observing the learning curve's shape and dynamics, DL models' behavior can be understood, and parameters can be tuned to increase performance. Fig. 10 presents accuracy and loss graph for VGG-16, VGG-19, and proposed custom CNN with augmentation and without augmentation. From the accuracy and loss graphs, it can be seen that there are fluctuations in the training and validation data. Models that run without data augmentation techniques tend to fluctuate more than models that run with data augmentation techniques. From Fig. 10(f), it is apparent that the accuracy of the training and validation gap is quite small, which is an indication of a well-fitting model. Additionally, even though the loss graph shows fluctuations, the actual losses are still considerably lower than those seen in other graphs. Therefore, based on the graphs of all of those models, the proposed model with data augmentation accuracy graph is the best fit among the others.

To examine the performance of CNN-based TL models and the proposed custom model, evaluation metrics such as accuracy, recall, precision, and f1-score are utilized. Table 5 demonstrates that models with applied augmentation techniques have a higher recall value than models without applied augmentation techniques. A considerable improvement in model performance has been observed with augmentation. VGG-16 without augmentation has achieved a recall of 87.00%, and with augmentation, it has achieved 92.00% recall. In VGG-19, there is an improvement of 7.00% in recall when applying augmentation techniques. However, custom CNN has always performed better than transfer-learning-based VGG-16 and VGG-19, regardless of whether

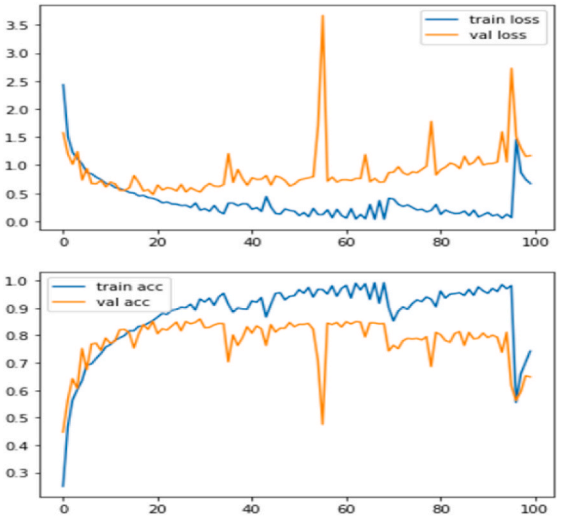
Fig. 9. Image of a random tomato leaf after applying augmentation techniques.



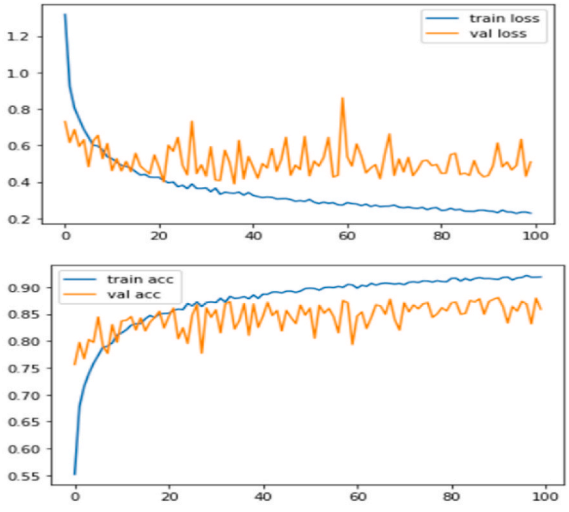
(a) Accuracy and loss graph for VGG-16.



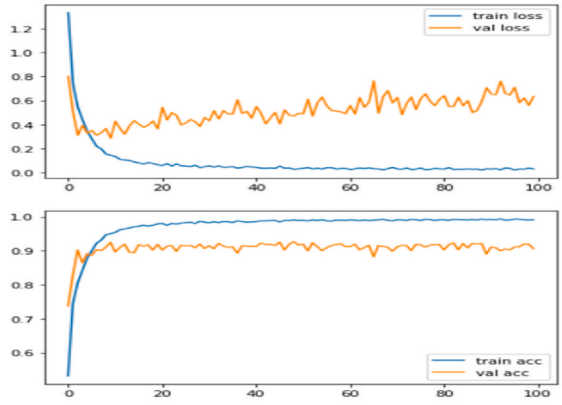
(b) Accuracy and loss graph for VGG-16 with augmentation.



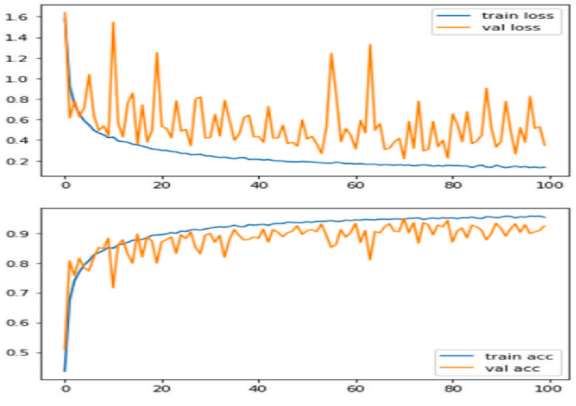
(c) Accuracy and loss graph for VGG-19.



(d) Accuracy and loss graph for VGG-19 with augmentation.



(e) Accuracy and loss graph for proposed custom CNN.



(f) Accuracy and loss graph for proposed custom CNN with augmentation.

Fig. 10. Accuracy and loss graph for VGG-16, VGG-19, and proposed custom CNN with augmentation and without augmentation.

augmentation techniques has applied or not. Among all the applied models, the proposed CNN model applied with data augmentation techniques has achieved the highest recall value of 95.00%.

The training and validation accuracy and loss compared to the final (test) accuracy of the applied models are presented in Table 6. By comparing models trained without augmentation to models trained with augmentation, a significant difference may be observed between training and validation loss and accuracy. Furthermore, by applying augmentation techniques, the applied models have achieved higher (test) accuracy compared to models without augmentation techniques. Among the many used models, the proposed model using augmentation techniques has achieved the highest accuracy of 95.00% with the

Table 5
Models evaluation metrics performance.

Models		Recall	Precision	F1-score
VGG-16 model	Without Augmentation	0.87	0.87	0.87
	With Augmentation	0.92	0.92	0.92
VGG-19 model	Without Augmentation	0.82	0.83	0.82
	With Augmentation	0.89	0.89	0.89
Proposed model	Without Augmentation	0.89	0.89	0.89
	With Augmentation	0.95	0.95	0.95

smallest loss and smallest accuracy difference between the training and validation datasets.

Table 7 demonstrates class-wise evaluation metrics generated by the VGG-16, VGG-19, and proposed custom models. In VGG-16, the recall value is higher when the augmentation technique is used than when it is not used. In a comparison of transfer-learning-based CNN models VGG-16 and VGG-19 with augmentation, VGG-16 outperformed VGG-19. In general, by applying augmentation techniques, the performance of all applied models has increased compared to models trained without augmentation. Among all the models, the proposed custom CNN model with augmentation demonstrated the most promising evaluation metrics values, which indicates a higher performance compared to other models.

The applied models' ROC-AUC curve is shown in Fig. 11. The VGG-19 model has the lowest AUC score compared to the other models. However, the VGG-19 with augmentation has performed better in terms of the AUC score and ROC curve than the VGG-19 without augmentation. Utilizing augmentation approaches has resulted in improved scores and more favorable ROC curves than models without augmentation techniques. The proposed model with augmentation achieves the highest AUC score. Moreover, the ROC curves of the proposed model with augmentation have attained the most optimal curves compared to other models.

Table 8 demonstrates some recent and related works on classifying the disease of the tomato leaf, which separately lists the applied architecture, publishing year, the considered class number and sample size, limitation, and the best-performing model with the obtained accuracy. To accomplish the classification of tomato leaf disease, a variety of CNN models, including transfer-learning-based CNN models and custom CNN, have been employed. In addition to CNN-based methods, ML learning models, Fuzzy SVM, and R–CNN have all been used to carry out

Table 6
Models training and validation accuracy and loss.

Model	Accuracy	Accuracy Training	Validation	Loss Training	Validation
VGG-16 model	87.00%	0.933	0.882	0.2116	0.4252
VGG-16 model (Augmentation)	92.00%	0.9135	0.9004	0.2420	0.3614
VGG-19 model	82.00%	0.8539	0.8508	0.4084	0.4768
VGG-19 model (Augmentation)	89.00%	0.8823	0.8694	0.3379	0.3895
Proposed model	89.00%	0.9503	0.9248	0.1442	0.2868
Proposed model (Augmentation)	95.00%	0.9464	0.9488	0.1558	0.2169

Table 7
Class-wise Recall, Precision, F1-score of various applied models.

	Class Name	Without Augmentation			With Augmentation		
		Recall	Precision	F1-score	Recall	Precision	F1-score
VGG-16 model	C0	0.92	0.85	0.88	0.89	0.96	0.92
	C1	0.78	0.83	0.81	0.86	0.87	0.86
	C2	0.83	0.85	0.84	0.90	0.85	0.87
	C3	0.87	0.83	0.85	0.93	0.94	0.93
	C4	0.79	0.77	0.78	0.89	0.91	0.90
	C5	0.77	0.96	0.85	1.00	0.78	0.88
	C6	0.89	0.79	0.84	0.81	0.93	0.87
	C7	0.93	0.99	0.96	0.96	0.98	0.97
	C8	0.92	0.99	0.95	0.97	0.99	0.98
	C9	0.94	0.95	0.95	0.97	0.97	0.97
	C10	0.88	0.73	0.79	0.93	0.94	0.94
VGG-19 model	C0	0.88	0.80	0.84	0.86	0.96	0.90
	C1	0.70	0.81	0.75	0.84	0.86	0.85
	C2	0.77	0.72	0.74	0.76	0.93	0.83
	C3	0.64	0.89	0.75	0.93	0.92	0.92
	C4	0.75	0.69	0.72	0.91	0.75	0.82
	C5	0.81	0.88	0.84	0.94	0.88	0.91
	C6	0.92	0.68	0.78	0.79	0.88	0.84
	C7	0.92	0.97	0.95	0.91	0.97	0.94
	C8	0.94	0.95	0.94	0.96	0.96	0.96
	C9	0.92	0.95	0.93	0.99	0.87	0.93
	C10	0.74	0.66	0.70	0.88	0.87	0.88
Proposed model	C0	0.88	0.87	0.88	0.94	0.94	0.94
	C1	0.85	0.88	0.86	0.94	0.92	0.93
	C2	0.87	0.81	0.84	0.93	0.94	0.93
	C3	0.89	0.93	0.91	0.96	0.97	0.97
	C4	0.82	0.84	0.83	0.92	0.94	0.93
	C5	0.95	0.93	0.94	0.97	0.96	0.96
	C6	0.89	0.96	0.92	0.87	0.96	0.91
	C7	0.95	0.98	0.97	0.98	0.98	0.98
	C8	0.92	0.98	0.95	0.97	0.97	0.97
	C9	0.94	0.96	0.95	0.99	0.92	0.95
	C10	0.82	0.60	0.69	0.92	0.93	0.92

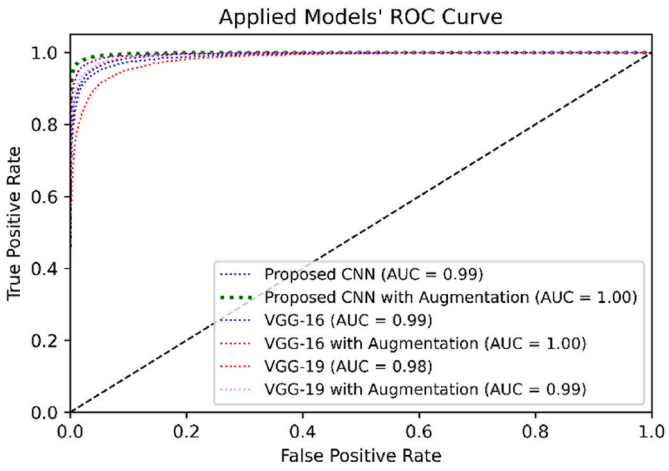


Fig. 11. ROC-AUC Curve of applied models.

the classification task. With best of our knowledge, a number of studies have classified nine diseases and one healthy category, but none have performed so for ten diseases and one healthy category. Furthermore, several studies have performed classification on a comparatively small dataset. Additionally, a variety of images that are captured or collected in various environments are present in our study's relatively large data set. In addition to eliminating the limitations of other factors, our study processed a relatively compact and efficient model that has higher accuracy compared to other research carrying out the same tomato leaf disease classification task. Furthermore, this study has deployed the best-preforming model to the web-based and developed a smartphone-based Android application for classifying tomato leaf disease.

4.5.2. *Deployment*

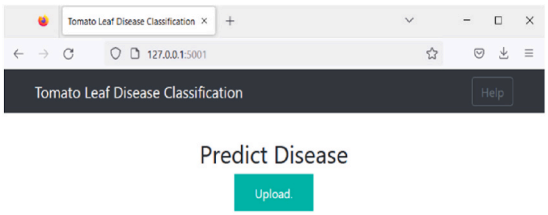
Deploying a DL model is the process of introducing a completed DL model into a production environment where it may be used for its intended purpose. The end goal of agriculture-based disease-related research is to benefit society and meet the requirements of the targeted mass audience. As tomato is grown all over the world and is vulnerable to a variety of diseases, early disease identification is essential for resolving the problem. Many farmers lack both the expertise to diagnose the disease and the resources to consult with a domain expert. As a result, they utilize a variety of pesticides and chemicals that can be detrimental to their crops and land while also placing a burden on their finances. Deploying an End-to-End (E2E) system that can assist tomato farmers is one of the primary objectives and motivations of this study.

For deploying the model to an E2E system, web-based and Android-based applications have been considered. Fig. 12 (a, b) illustrates the locally deployed tomato leaf disease system process and the result of the deployed model to predict tomato leaf disease from the perspective of the user. Fig. 12(c, d) depicts the user interface of the Android application as well as the live classification result. To predict tomato leaf disease based on an image, this study has implemented our most effective proposed model. This study has employed a variety of Python libraries and other supporting technologies to construct the web-based system. Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS) has been used to develop the website's frontend design. The backend mechanism has been constructed using the Python framework Flask. The user can upload an image of a tomato leaf, which will then be classified by the proposed model, and the result will be displayed on the screen.

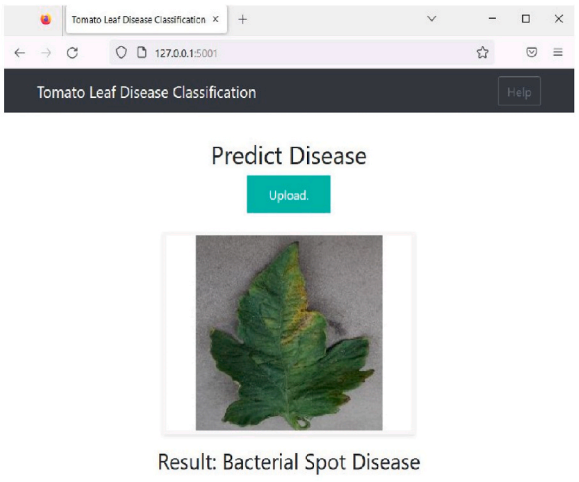
The user interface of an Android application has been developed using Extensible Markup Language (XML), while the backend is developed using Java. For image classification in the application, the best-performing proposed model has been converted to a lightweight TensorFlow version whose format type is "tflite". Then, the Java programming language has been utilized to implement this light TensorFlow model file in the Android application. The developed Android application is compatible with Android versions 5.0 to 12. The Android

Table 8
Result comparisons with related studies.

Reference	Year	Dataset Size	Number Of Class	Used Architecture	Best Model (Accuracy)	Limitation
TM et al. [20]	2018	18160	10	LeNet	LeNet (94.00–95.00%)	Single model applied, No deployment
Agarwal et al. [21]	2019	17500	10	VGG-16, InceptionV3, MobileNet, Custom CNN	Custom CNN (91.2%)	Lower accuracy, No deployment
Zhang et al. [22]	2018	5500	9	AlexNet, GoogLeNet, ResNet	ResNet with SGD (96.51%)	Lower number of class, Lower size of data, No deployment
Ahmad et al. [23]	2020	Laboratory 2364, field-based 15216	4, 6	VGG-16, VGG-19, ResNet, InceptionV3	InceptionV3 (99.60%)	Lower number of class, Lower size of data, No deployment
Basavaiah et al. [19]	2020	2000	5	DT, RT	RT (94.00%)	Lower size of data, No deployment
Chen et al. [26]	2020	8616	5	B-ARNet	B-ARNet (89.00%)	Lower accuracy, No deployment
Thangaraj et al. [27]	2020	16578	10	Modified-Xception	Modified-Xception (99.55%)	Single model applied, Lower number of class, No deployment
Zho et al. [28]	2021	4585	10	ResNet-50, SE-ResNet-50	SE-ResNet-50 (96.81%)	Lower size of data, No deployment
Kannan E et al. [29]	2020	12206	6	ResNet-50	ResNet-50 (97.00%)	Single model applied, Lower number of class, No deployment
Elhassouny and Smarandache [30]	2019	7176	10	MobileNet	MobileNet (90.3%)	Single model applied, Lower size of data, No deployment
Chen et al. [31]	2022	22390	10	AlexNet	AlexNet (98.00%)	Single model applied
H S and Sarojadevi H [32]	2022	735	7	Fuzzy-SVM, CNN, R–CNN	R–CNN (96.735%)	Lower number of class, Lower size of data, No deployment
Our proposed approach	–	32535	11	VGG-16, VGG-19, Proposed model	Proposed model (95.00%)	No real-time web-based deployment



(a) Image upload interface of Web application.



(b) Classification result in Web application.

(c) User Interface of Android application.

(d) Classification result in Android application.

Fig. 12. Tomato leaf disease classification applications.

application allows the user to classify tomato leaf disease by capturing a live image or selecting an existing image from the device. This study has provided open access for our developed smartphone-based Android application, which can be downloaded from the given GitHub repository [42]. The proposed model is lightweight and is able to predict the outcome and show results rapidly and accurately.

5. Conclusion

This study focuses on classifying tomato leaf disease by suggesting a custom CNN model and comparing that with CNN models based on TL. The TL-based models utilized are the pre-trained VGG-16 and VGG-19. The obtained dataset has been preprocessed and augmented. An ablation study has been carried out to determine the most appropriate augmentation methods and custom model layers and parameters. The proposed model's performance has been compared with the transfer-learning-based VGG-16 and VGG-19 models. Additionally, it is evident from the model evaluation that augmentation techniques have significantly increased the robustness of the models. The proposed CNN model has a higher accuracy of 95.00% among the employed models, making it significantly better than other research performing the same task. In addition, this study has deployed the most accurate classification model on a web- and Android based application for tomato leaf disease prediction.

In the future, findings from this research can serve as the state-of-the-art approach for future researchers who want to work with real-time classification tasks in the field of agriculture. Furthermore, the work's practical implications can be used for the classification of diseases on other plants' leaf surfaces. Various other techniques, such as segmentation, feature extraction, and ranking-based classification, can be combined with existing techniques to provide optimal results.

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.

References

[1] Picó B, Díez MJ, Nuez F. Viral diseases causing the greatest economic losses to the tomato crop. II. The Tomato yellow leaf curl virus — a review. Sci Hortic 1996;67(3–4):151–96. https://doi.org/10.1016/S0304-4238(96)00945-4.

[2] Lindhout P, Korta W, Cislik M, Vos I, Gerlagh T. Further identification of races of Cladosporium fulvum (Fulvia fulva) on tomato originating from The Netherlands France and Poland. Neth J Plant Pathol 1989;95(3):143–8. https://doi.org/10.1007/BF01999969.

[3] Kubota K, Tsuda S, Tamai A, Meshi T. Tomato mosaic virus replication protein suppresses virus-targeted posttranscriptional gene silencing. J Virol 2003;77(20):11016–26. https://doi.org/10.1128/JVI.77.20.11016-11026.2003.

[4] Tian M, Benedetti B, Kamoun S. A second kazal-like protease inhibitor from Phytophthora infestans inhibits and interacts with the apoplastic pathogenesis-related protease P69B of tomato. Plant Physiol 2005;138(3):1785–93. https://doi.org/10.1104/pp.105.061226.

[5] Chaerani R, Voorrips RE. Tomato early blight (Alternaria solani): the pathogen, genetics, and breeding for resistance. J Gen Plant Pathol 2006;72(6):335–47. https://doi.org/10.1007/s10327-006-0299-3.

[6] Gao W, Li B, Shi Y, Xie X. 'Studies on pathogenicity differentiation of Corynespora cassiicola isolates, against cucumber, tomato and eggplant. Acta Hortic Sin 2011;38(3):465–70.

[7] Dickey AM, Osborne LS, Mckenzie CL. Papaya (carica papaya , Brassicales: caricaceae) is not a host plant of tomato yellow leaf curl virus (Tylcv; family geminiviridae , genus Begomovirus). Fla Entomol 2012;95(1):211–3. https://doi.org/10.1653/024.095.0136.

[8] Abdulridha J, Ampatzidis Y, Kakarla SC, Roberts P. Detection of target spot and bacterial spot diseases in tomato using UAV-based and benchtop-based hyperspectral imaging techniques. Precis Agric 2020;21(5):955–78. https://doi.org/10.1007/s11119-019-09703-4.

[9] Foolad MR, Merk HL, Ashrafi H. Genetics, genomics and breeding of late blight and early blight resistance in tomato. Crit Rev Plant Sci 2008;27(2):75–107. https://doi.org/10.1080/07352680802147353.

[10] Tomato leaf mold. Available : https://extension.umn.edu/disease-management/tomato-leaf-mold. [Accessed 9 December 2022].

[11] Concepcion R, Lauguico S, Dadios E, Bandala A, Sybingco E, Alejandrino J. "Tomato septoria leaf spot necrotic and chlorotic regions computational assessment using artificial Bee colony-optimized leaf disease index,". In: Ieee region 10 conference (tencon); 2020. p. 1243–8. https://doi.org/10.1109/TENCON50793.2020.9293743. Osaka, Japan, Nov. 2020.

[12] Two-Spotted Spider Mite.Available : https://entomology.ca.uky.edu/ef310 [Accessed 9 December 2022].

[13] Tomato yellow leaf curl virus.Available : https://en.wikipedia.org/w/index.php?title=Tomato_yellow_leaf_curl_virus&oldid=1113057182 [Accessed 9 December 2022].

[14] Tomato mosaic virus. Available : https://en.wikipedia.org/w/index.php?title=Tomato_mosaic_virus&oldid=1046270277 [Accessed 9 December 2022].

[15] Powdery mildew on tomatoes. Available : https://blogs.cornell.edu/livegpath/gallery/tomato/powdery-mildew-on-tomatoes/[Accessed 9 December 2022].

[16] Blancard D. Tomato diseases. CRC Press; 2012. https://doi.org/10.1201/b15145.

[17] Schreinemachers P, Simmons EB, Wopereis MCS. Tapping the economic and nutritional power of vegetables. Global Food Secur 2018;16:36–45. https://doi.org/10.1016/j.gfs.2017.09.005.

[18] Tomato news. Available : https://www.tomatonews.com/[Accessed 15 November 2022].

[19] Basavaiah J, Arlene Anthony A. "Tomato leaf disease classification using multiple feature extraction techniques,". Wireless Pers Commun 2020;115(1):633–51. https://doi.org/10.1007/s11277-020-07590-x.

[20] Tm P, Pranathi A, SaiAshritha K, Chittaragi NB, Koolagudi SG. Tomato leaf disease detection using convolutional neural networks. In: Eleventh International Conference on Contemporary computing (IC3); 2018. p. 1–5. https://doi.org/10.1109/IC3.2018.8530532. Noida, Aug. 2018.

[21] Agarwal M, Singh A, Arjaria S, Sinha A, Gupta S. "ToLeD: tomato leaf disease detection using convolution neural network,". Proc Comput Sci 2020;167:293–301. https://doi.org/10.1016/j.procs.2020.03.225.

[22] Zhang K, Wu Q, Liu A, Meng X. Can deep learning identify tomato leaf disease? Adv Multimed 2018:1–10. https://doi.org/10.1155/2018/6710865. Sep. 2018.

[23] Ahmad I, Hamid M, Yousaf S, Shah ST, Ahmad MO. Optimizing pretrained convolutional neural networks for tomato leaf disease detection. Complexity 2020:1–6. https://doi.org/10.1155/2020/8812019. Sep. 2020.

[24] Mia MJ, Maria SK, Taki SS, Biswas AA. Cucumber disease recognition using machine learning and transfer learning. Bull EEI 2021;10(6):3432–43. https://doi.org/10.11591/eei.v10i6.3096.

[25] Biswas AA, Zulfiker Md S, Rajbongshi A, Mia Md J, Majumder A. Feature ranking based carrot disease recognition using MIFS method. In: Abraham A, Siarry P, Piuri V, Gandhi N, Casalino G, Castillo O, Hung P, editors. Hybrid Intelligent systems, vol. 420. Cham: Springer International Publishing; 2022. p. 56–68. https://doi.org/10.1007/978-3-030-96305-7_6.

[26] Chen X, Zhou G, Chen A, Yi J, Zhang W, Hu Y. Identification of tomato leaf diseases based on combination of ABCK-BWTR and B-ARNet. Comput Electron Agric 2020;178:105730. https://doi.org/10.1016/j.compag.2020.105730.

[27] Thangaraj R, Anandamurugan S, Kaliappan VK. Automated tomato leaf disease classification using transfer learning-based deep convolution neural network. J Plant Dis Prot 2021;128(1):73–86. https://doi.org/10.1007/s41348-020-00403-0.

[28] Zhao S, Peng Y, Liu J, Wu S. Tomato leaf disease diagnosis based on improved convolution neural network by attention module. Agriculture 2021;11(7):651. https://doi.org/10.3390/agriculture11070651.

[29] Kananan E N, Kaushik M, Prakash P, Ajay R, Veni S. Tomato leaf disease detection using convolutional neural network with data augmentation. In: 2020 5th International Conference on Communication and Electronics systems (ICCES). India: Coimbatore; 2020. p. 1125–32. https://doi.org/10.1109/ICCES48766.2020.9138030.

[30] Elhassouny A, Smarandache F. Smart mobile application to recognize tomato leaf diseases using Convolutional Neural Networks. In: 2019 International Conference of computer science and Renewable Energies (ICCSRE), Agadir, Morocco; 2019. p. 1–4. https://doi.org/10.1109/ICCSRE.2019.8807737.

[31] Chen H-C, et al. AlexNet convolutional neural network for disease detection and classification of tomato leaf. Electronics 2022;11(6):951. https://doi.org/10.3390/electronics11060951.

[32] Nagamani HS, Sarojadevi H. Tomato leaf disease detection using deep learning techniques. Int J Adv Comput Sci Appl 2022;13(no. 1). https://doi.org/10.14569/IJACSA.2022.0130138.

[33] Weight Initialization Schemes - Xavier (Glorot) and He. Available: https://mmuratarat.github.io//2019-02-25/xavier-glorot-he-weight-init [Accessed 9 December 2022].

[34] Activation Functions and their Derivatives – A Quick & Complete Guide. Available : https://www.analyticsvidhya.com/blog/2021/04/activation-functions-and-their-derivatives-a-quick-complete-guide/[Accessed 9 December 2022].

[35] Adam. Available : https://optimization.cbe.cornell.edu/index.php?title=Adam [Accessed 9 December 2022].

[36] Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names. Available : https://gombru.github.io/2018/05/23/cross_entropy_loss/[Accessed 9 December 2022].

[37] Keras Applications. Available : https://keras.io/api/applications/[Accessed 9 December 2022].

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

[39] Understanding the VGG19 Architecture. Available : https://iq.opengenus.org/vgg19-architecture/[Accessed 9 December 2022].

[40] Khan Q. Tomato disease multiple sources. 2022, October. Version 1. Available : https://www.kaggle.com/datasets/cookiefinder/tomato-disease-multiple-sources. [Accessed 28 October 2022].

[41] Confusion matrix.Available : https://en.wikipedia.org/w/index.php?title=Confusion_matrix&oldid=1107701525 [Accessed 9 December 2022].

[42] Showmick. Application-of-Tomato-Leaf-Disease-Classification [Android Package Kit]. 2023. https://github.com/showmick5/Application-of-Tomato-Leaf-Disease-Classification/tree/main.