frontiers | Frontiers in Plant Science

TYPE Original Research
PUBLISHED 17 May 2024
DOI 10.3389/fpls.2024.1382416

OPEN ACCESS

EDITED BY
Mohsen Yoosefzadeh Najafabadi,
University of Guelph, Canada

REVIEWED BY
Milind B. Ratnaparkhe,
ICAR Indian Institute of Soybean Research, India
Parvathaneni Naga Srinivasu,
Prasad V. Potluri Siddhartha Institute of Technology, India

*CORRESPONDENCE
Joe I. R. Praveen
praveen.joe@vit.ac.in

RECEIVED 05 February 2024
ACCEPTED 02 April 2024
PUBLISHED 17 May 2024

CITATION
V. P, Kumar AMS, Praveen JIR, Venkatraman S, Kumar SP, Aravintakshan SA, Abeshek A and Kannan A (2024) Improved tomato leaf disease classification through adaptive ensemble models with exponential moving average fusion and enhanced weighted gradient optimization. *Front. Plant Sci.* 15:1382416. doi: 10.3389/fpls.2024.1382416

Improved tomato leaf disease classification through adaptive ensemble models with exponential moving average fusion and enhanced weighted gradient optimization

Pandiyaraju V.[1], A. M. Senthil Kumar[1], Joe I. R. Praveen[1]*, Shravan Venkatraman[1], S. Pavan Kumar[1], S. A. Aravintakshan[1], A. Abeshek[1] and A. Kannan[2]

[1]School of Computer Science and Engineering, Vellore Institute of Technology, Chennai, India, [2]School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India

Tomato is one of the most popular and most important food crops consumed globally. The quality and quantity of yield by tomato plants are affected by the impact made by various kinds of diseases. Therefore, it is essential to identify these diseases early so that it is possible to reduce the occurrences and effect of the diseases on tomato plants to improve the overall crop yield and to support the farmers. In the past, many research works have been carried out by applying the machine learning techniques to segment and classify the tomato leaf images. However, the existing machine learning-based classifiers are not able to detect the new types of diseases more accurately. On the other hand, deep learning-based classifiers with the support of swarm intelligence-based optimization techniques are able to enhance the classification accuracy, leading to the more effective and accurate detection of leaf diseases. This research paper proposes a new method for the accurate classification of tomato leaf diseases by harnessing the power of an ensemble model in a sample dataset of tomato plants, containing images pertaining to nine different types of leaf diseases. This research introduces an ensemble model with an exponential moving average function with temporal constraints and an enhanced weighted gradient optimizer that is integrated into fine-tuned Visual Geometry Group-16 (VGG-16) and Neural Architecture Search Network (NASNet) mobile training methods for providing improved learning and classification accuracy. The dataset used for the research consists of 10,000 tomato leaf images categorized into nine classes for training and validating the model and an additional 1,000 images reserved for testing the model. The results have been analyzed thoroughly and benchmarked with existing performance metrics, thus proving that the proposed approach gives better performance in terms of accuracy, loss, precision, recall, receiver operating characteristic curve, and F1-score with values of 98.7%, 4%, 97.9%, 98.6%, 99.97%, and 98.7%, respectively.

KEYWORDS

deep learning, machine learning, image processing, ensemble learning, classification

1 Introduction

In the dynamic landscape of modern agriculture, where crop health plays a pivotal role in global food production, the precise and timely management of plant diseases is an ongoing challenge. Among these agricultural adversaries, leaf diseases emerge as intricate and multifaceted adversaries with distinct morphological manifestations. The science of leaf disease classification, a subdomain of plant pathology, is at the forefront of efforts to combat these detrimental afflictions. This research aspires to contribute to the field of leaf disease classification through the incorporation of pioneering technologies, namely, artificial intelligence (AI) and machine learning. The criticality of early detection and accurate classification in disease management cannot be overstated. Therefore, this study seeks to harness the potential of advanced algorithms, including convolutional neural networks (CNNs) and optimization into deep learning methodologies, to revolutionize the existing approaches to leaf disease diagnosis. At its core, this research addresses the challenges posed by leaf diseases by developing a novel classification system. By utilizing image recognition and deep learning techniques, this system aims to empower agriculture practitioners and plant pathologists with a sophisticated tool for disease identification. The impact of this system extends to many applications including crop health, reaching into the realms of global food security, sustainable agricultural practices, and environmental conservation.

Deep learning is an extension to the machine learning methods such as neural networks in AI that trains the computer system to recognize the patterns similar to the human brain. Deep learning models are trained to recognize even complex patterns found in images, text, videos, and voice data to perform accurate classifications and predictions. Deep learning algorithms perform both feature extraction and feature selection automatically without needing human effort as required in machine learning algorithms for training the software based on the algorithms. A CNN is one of the most important and fundamental deep learning neural network-based algorithms used for image recognition as it provides promising and accurate results in computer vision tasks. It has many architectural implementations including LeNet, AlexNet, Visual Geometry Group (VGG), GoogLeNet, and ResNet.

Time and space are important parameters to be considered for prediction-oriented decision-making systems. The temporal and spatial data on the disease growth in tomato leaves need time series analysis on image data with temporal reasoning. Moreover, prediction using time series analysis must focus on the direction of sequence that can be performed more effectively using machine learning-based classifiers. Moving average methods support to smoothen the time series analysis by identifying the temporal data patterns more effectively. Moreover, smoothing or filtering helps to eliminate the random variations that occur in the plotted time series data. An exponential (weighted) moving average method that applies a simple recursive procedure under the hood provides flexibility to the algorithm.

Despite the presence of many works on tomato plant leaf disease detection that are found in the literature, most of the existing systems use a machine learning approach for classification without any optimizer and temporal analysis. Therefore, it is necessary to employ manual preprocessing or to apply additional machine learning-based classification algorithms or clustering algorithms when performing effective feature extraction and feature selection. Moreover, the existing systems that use time series data are not designed to give higher importance to the most recent data and also do not focus on temporal reasoning by applying temporal constraints. Moreover, the convergence of the existing deep learning algorithm employed in the detection of tomato leaf diseases is not supported by an optimization algorithm. Finally, ensemble-based classification algorithms are not employed in the classification process to enhance the detection accuracy. Therefore, it is necessary to propose a new ensemble classifier with an optimization component and a temporal data analysis component.

In this paper, an ensemble model is proposed with an exponential moving average (EMA) function with temporal constraints based on interval analysis and an enhanced weighted gradient optimizer (EWGO) in which the gradient optimizer is enhanced with temporal rules and that is integrated into VGG-16 and Neural Architecture Search Network (NASNet) CNN architectures. VGG-16 is a fine-tuned model with a 16-layer depth developed by the VGG that consists of 13 convolution and max pooling layers with three fully connected layers, and it applies stride 2. The learning rate is fixed here as 0.1. The regression-based and binary classification-based loss functions are used in this work to reduce the errors. Moreover, the NASNet mobile training methods are integrated in this ensemble model for identifying the diseases in tomato leaves by providing improved learning and classification accuracy.

NASNet is also a CNN model that consists of two types of cells, namely, the normal and the reduction cells. The EMA method is used in this ensemble model since it gives more weightage to the current data in the temporally oriented time series data. Moreover, the Plant Village dataset is used in this work to carry out the experiments for testing the ensemble model proposed in this paper. Moreover, the Plant Village dataset is a publicly available dataset consisting of 54,305 images from which 1,000 images related to tomato leaves have been extracted and used in this work for training and testing the system. The main advantages of the proposed ensemble model are the increase in classification accuracy and the reduction in error rate in the detection of tomato leaf diseases.

The main motivation for this research work is that the profession of agriculture is one of the most vital in every world economy. It is the main source of resources in our country. Nowadays, leaf disease has a great impact on the productivity of vegetables. If we cannot control the disease, then it can greatly affect the harvest. These problems provide great motivation in finding out the origin of the disease at an earlier stage to help the tomato plants grow healthily and increase their yield. Another motivation for this research is that it addresses the challenges posed by leaf diseases by developing a novel classification system. By utilizing image recognition and deep learning techniques, this system aims to empower agriculture practitioners and plant pathologists with a sophisticated tool for disease identification. The impact of this

system extends beyond crop health, reaching into the realms of global food security, sustainable agricultural practices, and environmental conservation.

In this work, the Plant Village dataset is used to carry out the experiments for testing the model proposed in this paper. Moreover, the Plant Village dataset is a publicly available dataset consisting of 54,305 images from which 1,000 images related to tomato leaves have been extracted and used in this work for training and testing the system. The Plant Village dataset provides data to detect 39 different plant diseases. Moreover, the dataset contains 61,486 images of plant leaves with backgrounds. The dataset was designed using six different augmentation techniques in order to create more diverse datasets with different background conditions. The augmentations that have been used in this process include scaling, rotation, injection of noise, gamma correction, image flipping, and principal component analysis to perform color augmentation.

The main contributions of this paper are as follows:

- Proposal of an ensemble model using VGG-16 and NASNet mobile training deep learning models with an EMA function.
- Effective time series analysis using the CNN-based deep learning classifier along with an EWGO.
- Use of the Plant Village dataset for validation.
- Evaluation using suitable metrics.

The research unfolds in the following sequence: Section 2 provides a comprehensive exploration of the taxonomy and intricacies of leaf diseases. Section 3 is a detailed methodology section highlighting the technical aspects of image processing and machine learning, and the revelation of a state-of-the-art deep learning classification system designed to improve the accuracy and efficiency of leaf disease identification. In section 4, performance assessment of the proposed approach and results are compared with existing techniques. We conclude the research paper in section 5.

The VGG-16 architecture is a deep CNN designed for image classification tasks. It was introduced by the VGG at the University of Oxford. VGG-16 is characterized by its simplicity and uniform architecture, making it easy to understand and implement.

2 Literature survey

There are many works on tomato leaf detection, machine learning (Uma et al., 2016; Anusha and Geetha, 2022; Harakannanavara et al., 2022), deep learning (Haridasan et al., 2023; Sankareshwaran et al., 2023; Yakkundimath and Saunshi, 2023), optimization techniques, data mining (Das and Sengupta, 2020; Demilie, 2024), regression analysis, image analysis (Ganatra and Patel, 2020; Ngugi et al., 2021), and prediction techniques that are found in the literature. Mustafa et al. (2023) proposed a five-layer CNN model for detecting plant diseases using leaf images. A total of 20,000 images were used to train the model. This model detects the pepper bell plant leaf disease with better accuracy. The results are evaluated in terms of accuracy, precision, and recall, and F1-scores are computed. The model performs better than state-of-the-art models. Seetharaman et al. (Seetharaman and Mahendran, 2022) presented a region-based CNN model to detect a banana leaf disease using Gabor extraction. Images are preprocessed by histogram pixel localization with media filter. The segmentation part is done with region-based edge normalization. Feature extraction is performed using the novel method Gabor-based binary patterns with CNN. A region-based CNN helps in detecting the disease area. The results are evaluated and they perform better than CNN, DCNN, ICNN, and SVM models in terms of precision, recall, accuracy, and sensitivity.

Nerkar et al. (Nerkar and Talbar, 2021) proposed a method to detect leaf disease using a two-level nonintrusive method. This model combines generative adversarial network and reinforcement learning. Cross dataset learning is used. CNN is combined with GAN for better results. Re-enforcement learning retrains the GAN using confidence scores. Classification results are evaluated and results are higher than other models. Mukhopadhyay et al. (2021) proposed a non-dominated sorting genetic algorithm for tea leaf disease detection. Image clustering is the main idea of this model. PCA is used for feature reduction and multi-class SVM is used for disease detection. Five various datasets of tea leaf are used in the work. The proposed model provides better accuracy than traditional models.

Vallabhajosyula et al. (2022) proposed a transfer learning-based neural network for plant leaf disease detection. In this work, pre-trained models were used. The deep ensemble neural network is used along with pre-trained models. Transfer learning and data augmentation are used for parameter tuning. The results are evaluated and provide higher accuracy with lesser number of computations. Huang et al. (2023) discussed a tomato leaf disease detection model using the full convolutional neural network (FCN) with suitable normalization dual path networks. The FCN used to segment the target crop images and improve the dual path network model is used for feature extraction. The results are evaluated on the augmentation dataset and accuracy is better than other models.

Chouhan et al. (2021) proposed a model for leaf disease detection using the fuzzy-based function network. Initially, preprocessing is done and the scale-invariant feature transform method is used for feature extraction. The fuzzy-based function network is used for detecting the leaf disease. Training is done with the help of the firefly algorithm. The model results are evaluated in terms of accuracy and are higher than traditional models. He et al. (2023) presented a maize leaf disease detection model using machine vision. The batch normalization layer is appended with the convolution layer to fasten the convergence speed of the network. Cost function is developed to increase the detection accuracy. Four types of pre-trained CNN models are used for feature extraction network for training. The gradient descent algorithm is applied to optimize the model performance. The results are evaluated in terms of F1-score, recall rate, and accuracy.

Ruth et al. (2022) proposed a deep learning model for disease detection using the meta-heuristic algorithm. CNN is used for feature extraction. The optimal deep neural network is used for disease detection. A two-level weight optimization is used to

increase the performance of the detection model. Two-level weight optimization is achieved using an improved butterfly optimization algorithm, where the genetic algorithm is used to improve the butterfly optimization algorithm. The results are evaluated in terms of sensitivity, accuracy, and specificity. The overall accuracy is higher than other traditional models. Andrushia et al. (Andrushia and Patricia, 2020) presented a leaf disease detection model using the artificial bee colony optimization algorithm. Initially, preprocessing is done by removing noises and background images. Shape, color, and texture are extracted as features and are sent to the support vector machine model for disease detection. The model results are better in terms of recall, precision, and accuracy.

Abed et al. (2021) presented a novel deep learning model for bean leaf disease detection. This model contains two phases: detection and diagnosing. For detection, the U-Net architecture using the ResNet34 encoder is used. In the classification part, results are evaluated for five different deep learning models. The dataset contains 1,295 images of three classes such as healthy, bean rust, and angular leaf spot. The results are evaluated in terms of sensitivity, specificity, precision, F1-score, and area under the curve (AUC). Pandey et al. (Pandey and Jain, 2022) proposed a deep attention residual network using an opposition-based symbiotic organisms search algorithm. In this model, residual learning blocks are used with the attention learning mechanism for feature extraction. A new CNN model, AResNet-50, is designed for disease detection. The opposition-based symbiotic organisms search algorithm is used to tune the parameters of the model. Plants like citrus, guava, eggplant, and mango leaves are considered for the experimental analysis. The results of the model are evaluated in terms of accuracy, and they are better than those of the existing models such as AlexNet, ResNet-50, VGG-16, and VGG-19. Zhao et al. (2020) proposed a multi-context fusion network model for crop disease detection. In this model, standard CNN is used to extract visual features from 50,000 crop disease samples. Contextual features are collected from image acquisition sensors. A deep, fully connected network is proposed by combining contextual features and visual features to detect the leaf disease. The model performance is evaluated in terms of accuracy, which is higher than state-of-the-art methods.

Wang et al. (2017) proposed a new technique for automatic estimation of plant disease severity using image analysis through the effective application of deep learning algorithms. Bracino et al. (2020) explained the development of a new hybrid model based on machine learning techniques for the accurate detection of health using disease classification. Ashwinkumar et al. (2022) proposed an automated plant leaf disease detection model using deep learning classification named optimal MobileNet, which is designed based on CNNs. Khan et al. (2019) developed one optimized method for disease detection using image segmentation and classification for identifying the apple diseases. The authors made the decisions by analyzing whether there is a strong correlation among the features and also using genetic algorithm for feature selection. Most of the works found in the literature on tomato leaf disease detection used the benchmark dataset, namely, the Plant Village dataset (Kaustubh, 2020).

Sanida et al. (2023) proposed a new methodology for the effective detection of tomato leaf diseases by identifying them using a two-stage transfer learning model. Pandiyaraju et al. (2023) proposed an optimal energy utilization technique for reducing the energy consumption via the agricultural sensors used in precision agriculture. These sensors have been connected to a WSN that performs energy optimization by using a multi-objective clustering and deep learning algorithm to reduce the energy consumption. In another related work, Pandiyaraju et al. (2020) developed an energy-efficient routing algorithm for WSNs using clustering of nodes. Moreover, the routing decision has been made in their work using intelligent fuzzy rules that were applied in precision agriculture. In the area of agriculture and gardening, Pandiaraju et al. (Pandiyaraju et al., 2017) proposed a rule-based intelligent roof control algorithm for effective water conservation without affecting the agricultural yield with respect to smart terrace gardening. Such a model can be enhanced to detect the leaf diseases for providing better yield with minimum water.

Shoaib et al. (2023) presented a review of deep learning classification algorithms that have been used in the detection of plant leaf diseases. Santhosh et al. (2014) proposed a farmer advisory system using intelligent rules based on machine learning classifier. Jabez Christopher et al. (Jabez et al., 2015) proposed an optimized classification model that uses rules based on knowledge mining with swarm optimization for providing effective disease diagnosis. Gadade et al. (Gadade and Kirange, 2022) proposed an intelligent approach based on deep learning for the effective detection of tomato leaf diseases from leaf images that have captured with varying capturing conditions. Saeed et al. (2023) proposed one new smart detection methodology for the accurate detection of tomato leaf diseases by using transfer learning-based CNNs. Shoaib Muhammad et al. (Shoaib et al., 2022) proposed a new model for tomato leaf disease detection by using deep learning algorithms for performing both segmentation and classification of leaf images.

Sreedevi and Manike (2024) presented a new solution for identifying the tomato leaf disease based on classification using a modified recurrent neural network through severity computation. Prabhjot Kaur et al. (2024) carried out a performance analysis on the image segmentation models that are used to detect leaf diseases present in the tomato plants. Thai-Nghe et al. (Nguyen et al., 2023) presented a deep learning-based approach for the effective detection of tomato leaf diseases. Chang et al. (2024) developed one general-purpose edge-feature-guided model for the identification of plant diseases by enhancing the power of vision transformers. Li et al. (2023) presented a new lightweight vision transformer model based on shuffle CNNs for the effective diagnosis of leaf diseases in sugarcane plants. Thai et al. (2023) proposed a new vision transformer model designed for the accurate detection of cassava leaf diseases.

Yu et al. (2023) explained the use of inception convolutional vision transformers for the effective identification of plant diseases. Arshad et al. (2023) developed an end-to-end and hybrid model based on the deep learning framework for the accurate prediction of potato leaf diseases. Shiloah et al. (Elizabeth et al., 2012) proposed one new segmentation approach based on machine learning model for improving the diagnostic accuracy of detecting lung cancers from chest computed tomography images. Dhalia Sweetlin et al. (2016) proposed a patient-specific model for the effective

segmentation of lung computed tomographic images. Singh and Misra (2017) proposed a machine learning-based model for the effective detection of plant leaf diseases by performing suitable image segmentation. Agarwal et al. (2020) developed a new system for tomato leaf disease detection by applying the CNN classifier.

Chen et al. (2022) proposed the use of the AlexNet CNN model for the effective detection of tomato leaf diseases by performing accurate classification of tomato leaf images. Ganapathy et al. (2014) proposed an intelligent temporal pattern classification model by using fuzzy temporal rules with particle swarm optimization algorithm. Jaison et al. (Bennet et al., 2014) proposed a discrete wavelet transform-based feature extraction model along with one hybrid machine learning classification algorithm for performing effective microarray data analysis. Elgin Christo et al. (2019) proposed a new correlation-based ensemble feature selection algorithm that has been developed using bioinspired optimization algorithms integrated with a backpropagation neural network-based classifier.

Thangaraj et al. (2021) proposed an automated tomato leaf disease classification algorithm by using a transfer learning-based deep CNN classifier. Al-Gaashani et al. (Al-gaashani et al., 2022) proposed a new model for tomato leaf disease classification by the application of transfer learning with feature concatenation. Han et al. (2017) proposed a new weighted gradient-enhanced classification model not only to provide high-dimensional surrogate modeling but also to perform design optimization. Wu et al. (2021) proposed a new distributed optimization method that uses weighted gradients for solving the economic dispatch problem pertaining to the multi-microgrid systems. Abouelmagd et al. (2024) developed an optimized capsule neural network for the effective classification of tomato leaf diseases. Other approaches that are used in the detection of leaf diseases include those with deep learning and also with explainable AI (Rakesh and Indiramma, 2022; Bhandari et al., 2023; Debnath et al., 2023; Nahiduzzaman et al., 2023).

Despite the presence of all these related work in the literature, most of the segmentation and classification algorithms use a machine learning approach for classification. Therefore, it is necessary to employ either manual work or additional classification algorithms for performing feature extraction and feature selection. Moreover, the time series data are not analyzed by giving higher importance to the most recent data by the application of temporal constraints. The convergence of the existing deep learning algorithm employed in the detection of tomato leaf diseases is not supported by an optimization algorithm. Finally, ensemble-based classification algorithms are not employed in the classification process to enhance the detection accuracy. In order to handle all these limitations that are present in the existing systems developed for accurate tomato leaf disease detection, a new ensemble classification model is proposed in this paper that uses an EMA function with temporal constraints, and it is supported by an EWGO along with fine-tuned VGG-16 and NASNet mobile training methods for enhancing the classification accuracy that can increase the detection accuracy with respect to the detection of tomato leaf diseases.

3 Proposed work

3.1 Method

The data that show the features are initially analyzed using histogram plots and pie charts for better visualization of the data statistics to check for data imbalance among different classes. It has been concluded via complete exploration that there is no data imbalance and that the features of the images have been completely studied.

Next, the images are preprocessed in order to enhance the learning ability of our deep learning models. A median filter is applied on the image to remove noise to improve image quality. Redundant parts of the image that do not contribute to the model's learning process are also removed. Furthermore, the α and β factors in our images are adjusted in order to modify the brightness and contrast, thereby making the region of interest more prominent. The images are finally normalized to have pixel values ranging from 0 to 1, and the data are augmented to ensure a wider scale of learning by the model.

For the initial part of feature extraction, the VGG-16 transfer learning model undergoes fine-tuning by unfreezing its last five layers, enabling to adapt the model that originally contained ImageNet's weights to the specified dataset. By employing the use of Global Average Pooling to pool the CNN layers' features, the data are then passed into two fully connected layers ultimately leading to the output layer. The optimization of the model is achieved using the Adam optimizer with a learning rate of 0.0001, and evaluation metrics such as the F1-score, AUC score, precision, and recall are applied.

The NASNet mobile transfer learning model is employed with ImageNet weights for the next part. A flattened layer is then used to transform the outputs from the CNN layers into a one-dimensional tensor that facilitates the passage through three fully connected layers that ultimately reach the output layer. The optimization of the model is once again achieved using the Adam optimizer with a learning rate of 0.0001, and evaluation metrics such as the F1-score, AUC score, precision, and recall are applied.

The extracted features obtained from the two transfer learning models are now taken and passed on as parameters to a custom ensemble layer that incorporates EMA function that emphasizes the recent data points with greater weights. The resulting ensemble model shows an optimized learning curve by adopting the adaptive rate of learning, which is achieved by using a custom EWGO that modifies the learning rate based on custom ensemble weight suitable for our custom ensemble model.

3.2 Dataset

This research utilizes the dataset (Kaustubh, 2020) that consists of a collection of tomato leaf images, each belonging to one of nine distinct categories, representing various leaf diseases or a healthy state (no disease). The dataset encompasses a total of 10,000 images designated for training and an additional 1,000 images reserved for

testing. To facilitate model development and evaluation, we partitioned the training dataset into a 75%–25% split, resulting in 7,500 images allocated for training and 2,500 images for validation, and the entire additional 1,000 images were reserved for the test set.

This dataset serves as the foundation for the development of the proposed model, which aims to enhance the classification of tomato leaf diseases.

3.3 Preprocessing

The following are the steps involved in preprocessing:

- Median filter
- Image cropping
- Brightness and contrast adjustments
- Normalization

3.3.1 Median filter

The first step of data preprocessing utilizes a median filter, which is a non-linear digital image filtering technique that runs through the signal as one entry after another by replacing the entry value by the median of the neighboring entry values, which depends on the window size, resulting in the removal of the salt-and-pepper noise in an image. In this case, a window size of 3 has been chosen for preprocessing the image.

This median filter is represented mathematically as shown in Equation (1):

$$g(x,y) = Med(f(x,y)) \tag{1}$$

where $f(x,y)$ is the window array and $g(x,y)$ is the median value of the window array. The steps for the median filter are shown in Algorithm 1.

```
function median_filter():
    input: raw tomato_leaf_image;
    output: median_filtered_image;
    image = input;
    l = length of image;
    b = breadth of image;
    c = channels of image;
    w = window_size;
    filtered_image = create_empt;y_image(l,b)
    b_image = img[l][b][1];
    g_image = img[l][b][2];
    r_image = img[l][b][3];
    for i = 0 to l − 1 do:
        for j = 0 to b − 1 do:
            b_img = image[i][j][1]
            g_img = image[i][j][2];
            r_img = image[i][j][3];
        end for
    end for
    apply_median_filter(b_img,w);
    apply_median_filter(g_img,w);
    apply_median_filter(r_img,w);
    for i = 0 to l − 1 do:
        for j = 0 to b − 1 do:
            filtered_image = [b_img[i][j], g_img[i][j], r_img[i][j]];
        end for
    end for
end function
End
Function apply_median_filter():
    input: single_channel_tomato_leaf_image;
    output: median_filtered_single_channel_image;
    len = length of img;
    bt = length of img;
    applied_img = create_array(len,bt);
    wh = w/2 ;
    for x = 0 to len − 1 do:
        for b = 0 to bt − 1 do:
            window = [];
            for i = −wh to wh − 1 do:
                for j = −wh to wh − 1 do:
                    winx = x + i;
                    winy = y + j;
                    if winx >= 0 and winy >= 0 and winx <
                    len and winy < bt then:
                        append value to window
                        (img[winx][winy])
                    end if
                end for
            end for
        end for
    end function
End
```

Algorithm 1. Median filter.

3.3.2 Image cropping

Since the outer areas of the image are not helpful with the tomato disease detection, the size of the image is reduced by 10 pixels on each side, thus reducing the image size from 256 × 256 to 236 × 236 by removing the areas where there are no significant features for disease detection. The steps for image cropping are shown in Algorithm 2.

```
Function crop_image ():
    input: median_filtered_image
    output: cropped_median_filtered_image
    img = median filtered image
    length = length of img
    breadth = breadth of img
    crop_value = 10
    max_crop_length = length − crop_value
    max_crop_breadth = breadth − crop_value
    crop_image = create empty image of dimensions(max_
    crop_length,max_crop_breadth)
```

```
        crop_image = img[crop_value:max_crop_length][crop_
        value:max_crop_breadth]
    end Function
end
```

Algorithm 2. Image cropping.

3.3.3 Brightness and contrast enhancements of images

For better-quality images and improved ability of the CNN to identify the region of interest, its brightness is reduced and the contrast of the image is increased. This mitigates overexposure of the images, allowing the CNN to extract the features in the region of interest easily due to better visibility.

Brightness and contrast enhancement can be represented mathematically as shown in Equation (2):

$$g(i,j) = \alpha f(i,j) + \beta \tag{2}$$

where α is the contrast factor and β is the brightness factor. $f(i,j)$ represents the pixel of the input image, which is the cropped image, while $g(i,j)$ is the output image where the image's brightness and contrast are adjusted using α and β. The procedure for brightness and contrast enhancements is shown in Algorithm 3.

```
Function adjust_image():
    input:cropped_median_filtered_image,brightness_
    factor,contrast_factor
    output:cropped_filtered_image_with_adjustments
    image = cropped_median_filtered_image
    l ← length of image
    b ← breadth of image
    c ← channels of image
    adjusted_img ← create empty image of dimensions
    l and b
    α ← contrast_factor
    β ← brightness_factor
    for i = 0 to l − 1 do:
        for j = 0 to b − 1 do:
            for k = 0 to c − 1 do:
                adjusted_img[i][j][k] ← α*image[i][j][k] + β
            end for
        end for
    end for
end Function
end
```

Algorithm 3. Brightness and contrast enhancement.

3.3.4 Image normalization

For better weight initialization and to maintain consistency in the pixel range of the input, the image is normalized so that all pixel values are confined to the interval [0, 1]. Due to this, the deep learning model's convergence is enhanced with the range reduction from 255 to 1 by dividing each pixel value by 255. This process also improves the learning rate of our proposed model and the stability of the model during training. The procedure for image normalization is shown in Algorithm 4.

```
Function normalize_image():
    input:brightness_and_contrast_adjusted_image
    output:normalised_image
    image = input
    l ← length of image
    b ← breadth of image
    c ← channels of image
    normalisation_value ← 255
    normalised_image ← create_empty_image(l,b)
    for i = 0 to l − 1 do:
        for j = 0 to b − 1 do:
            for k = 0 to c − 1 do:
                normalised_image[i][j][k] ← image[i][j][k]/255
            end for
        end for
    end for
end Function
end
```

Algorithm 4. Image normalization.

3.4 Feature extraction and classification

Upon successful completion of preprocessing, the tomato leaf images are subjected to appropriate feature extraction and thereby will be classified using the deep learning model. This, in turn, will support not only the identification of diseases in the leaves but also the severity. The deep learning model used is the VGG-16 fine-tuned model. In addition, a CNN model, namely, NASNet, is also employed for the leaf's disease identification.

Later, an ensemble model consisting of five ensemble blocks and a final output block is used with the input layer being received from the output of the VGG-16 fine-tuned model and the NASNet model as a list. Furthermore, the results are improved for an enhanced performance with the aid of an EMA-based approach and optimized with an EWGO.

3.4.1 VGG-16 fine-tuned model

The last five layers of the VGG-16 model are unfrozen and the weighs of these layers are updated with the data to fine-tune the model. The optimizers do not modify the parameters of the remaining layers, which remain frozen, thereby preserving the weights.

This model, which is made up of five different blocks, is composed of convolution layers with rectified linear unit (ReLU) activation and a max pooling layer, a global average pooling layer, dense layers, batch normalization layers, and an output dense layer with softmax activation. The preprocessed image of size 236 × 236 × 3 is taken as an input into the model, first entering block 1.

Block 1 consists of two convolution layers and a max pooling layer. Each convolution layer consists of 64 filters, each of size 3 × 3. Each layer also has a ReLU activation layer that brings in non-linearity once the feature extraction is done by that layer. The first convolution layer receives the input as 236 × 236 × 3, and the first convolution layer produces the output of shape 236 × 236 × 64 after the activation function. The second convolution layer takes the input as the output of the first convolution layer and performs feature extraction and ReLU activation without making any changes in the shape of the data. Once the output data are produced by the second convolution layer, the max pooling layer that has a filter size of 2 × 2 reduces the size from 236 × 236 × 64 to 118 × 118 × 64, which sends the output to block 2.

Block 2, just like block 1, consists of two convolution layers where each layer has a ReLU activation function and a max pooling layer. The only difference is that the input received by the first convolution layer of this block will be of size 118 × 118 × 64. At the end of the second convolution, the output will be of size 118 × 118 × 128 since the number of filters in the convolution layers of the second block is 128. The max pooling layer reduces the size of the data from 118 × 118 × 128 to 59 × 59 × 128.

Block 3, unlike the previous two blocks, has three convolution layers where each layer has a ReLU activation function and a max pooling layer. The functionality of the block remains the same with the difference here being the presence of a third convolutional layer and the presence of 256 filters in each convolution layer. The first convolution layer receives the input of size 59 × 59 × 128 from the max pooling layer of block 2 and produces an output of size 59 × 59 × 256, which is preserved in the second and third convolution layer. The max pooling layer reduces the size of the data to 29 × 29 × 256.

Blocks 4 and 5 are similar to block 3 with the only difference being all the convolution layers present in blocks 4 and 5 have 512 filters. The input received by the first layer of block 4 will be of dimension 29 × 29 × 256 and the output after the third convolution layer will be of size 29 × 29 × 512, which, in turn, is reduced to 14 × 14 × 512 by the max pooling layer. In case of block 5, the input received by the first convolution layer will be of size 14 × 14 × 512 and the output is preserved even after the third convolution layer. The max pooling layer in block 5 reduces its size from 14 × 14 × 512 to 7 × 7 × 512.

The global average pooling layer takes the output of block 5 as input, which down-samples the multi-dimensional data into single-dimensional data by finding the average of each feature map where the filter is of size 2 × 2, resulting in the reduction of data size from 7 × 7 × 512 to 1 × 1 × 512. After this down-sampling, two dense layers with ReLU activation composed of 128 and 32 neurons, respectively, transform the output obtained by extracting the features of the preceding layers into data, which are suitable for classification. Finally, the output layer, i.e., dense layer with softmax activation, is used to perform multiclass classification. The steps for VGG-16 fine-tuned model is shown in Algorithm 5.

input: *preprocessed tomato leaf image*
output: *trained finetuned_VGG16 classifier for tomato leaf disease classification*

Function *TrainClassifier(preprocessed_tomato_leaf_image):*

- ***model*** ← *VGG16 multiclass Classifier*
  - ***k*** ← *finetuneable layers*
- ***for*** *layer in last k model layers* ***do***
  - ***layer*** ← *trainable*
- ***end for***
- β ← *batch size*
- N ← *total classes of tomato leaf diseases*
- h ← *height of preprocessed_tomato_leaf_image*
- w ← *width of preprocessed_tomato_leaf_image*
- c ← *color channels of preprocessed_tomato_leaf_image*
  - ***for*** *epoch* = 1 *to* 100 ***do***
    - μ ← *learning rate*
    - ***while*** *performance does not plateau* ***do***
    - ***batch*** ← *obtain a batch of size* β
    - *feed* ***batch*** *into model through layers* L
    - ***prob*** ← *predicted tomato leaf disease class probabilities*
    - ***labels*** ← *ground truth probabilities*
    - *loss,* δ ← *categorical cross entropy loss*

$$\delta \leftarrow -\log\left(\frac{e^{x_p}}{\sum_{j=1}^{N} e^{x_j}}\right)$$

$$x_i \leftarrow \textit{logit for class } i \in \{1, 2, \ldots, N\}$$

    - *update model parameters* θ *through backpropagation using loss* δ

$$\theta \leftarrow \theta - \mu \nabla \delta$$

    - *where* $\nabla\delta$ ← *gradient of loss* δ *with respect to model parameters* θ
    - *compute* ***accuracy***,

$$accuracy = \frac{\sum_{k=1}^{N}(TP_k + TN_k)}{\sum_{k=1}^{N}(TP_k + TN_k + FP_k + FN_k)}$$

    - *compute* ***precision***,

$$precision = \frac{\sum_{k=1}^{n} TP_k}{\sum_{k=1}^{n}(TP_k + FP_k)}$$

    - *compute* ***recall***,

$$recall = \frac{\sum_{k=1}^{n} TP_k}{\sum_{k=1}^{n}(TP_k + FN_k)}$$

    - *compute* $F1$ – ***score***,

$$F1 - Score = \frac{2\sum_{k=1}^{n} TP_k}{\sum_{k=1}^{n}(2TP_k + TN_k + FP_k)}$$

    - *use* ***Adam*** *optimizer to monitor loss* δ *and tune model learning*;
    - ***end while***
    - ***if*** *performance plateaus* ***then***

update learning rate μ to promote further learning

end if

end for

$output_{VGG16} \leftarrow$ output probabilities from model

return $output_{VGG16}$

end Function

end

Algorithm 5. Tomato leaf classification—fine-tuned VGG-16 training.

input: preprocessed tomato leaf image

output: trained NASNet classifier for tomato leaf disease classification

Function TrainClassifier(preprocessed_tomato_leaf_image):

$model \leftarrow$ NASNet multiclass Classifier

$\beta \leftarrow$ batch size

$N \leftarrow$ total classes of tomato leaf diseases

$h \leftarrow$ height of preprocessed_tomato_leaf_image

$w \leftarrow$ width of preprocessed_tomato_leaf_image

$c \leftarrow$ color channels of preprocessed_tomato_leaf_image

for epoch = 1 to 100 ***do***

$\mu \leftarrow$ learning rate

while performance does not plateau ***do***

$batch \leftarrow$ obtain a batch of size β

feed ***batch*** into ***model*** through layers L

$prob \leftarrow$ predicted tomato leaf disease class probabilities

$labels \leftarrow$ ground truth probabilities

loss, $\delta \leftarrow$ categorical cross entropy loss

$$\delta \leftarrow -\log\left(\frac{e^{x_p}}{\sum_{j=1}^{N} e^{x_j}}\right)$$

$x_i \leftarrow$ logit for class $i \in \{1, 2, \ldots, N\}$

update model parameters θ through back propagation using loss δ

$$\theta \leftarrow \theta - \mu\nabla\delta$$

where $\nabla\delta \leftarrow$ gradient of loss δ with respect to model parameters θ

compute ***accuracy***,

$$accuracy = \frac{\sum_{k=1}^{N}(TP_k + TN_k)}{\sum_{k=1}^{N}(TP_k + TN_k + FP_k + FN_k)}$$

compute ***precision***,

$$precision = \frac{\sum_{k=1}^{n} TP_k}{\sum_{k=1}^{n}(TP_k + FP_k)}$$

compute ***recall***,

$$recall = \frac{\sum_{k=1}^{n} TP_k}{\sum_{k=1}^{n}(TP_k + FN_k)}$$

compute $F1 - score$,

$$F1 - Score = \frac{2\sum_{k=1}^{n} TP_k}{\sum_{k=1}^{n}(2TP_k + TN_k + FP_k)}$$

use ***Adam*** optimizer to monitor loss δ and tune model learning;

end while

if performance plateaus ***then***

update learning rate μ to promote further learning

end if

end for

$output_{NASNet} \leftarrow$ output probabilities from ***model***

return $output_{NASNet}$

end Function

end

Algorithm 6. Tomato leaf classification—NASNet training.

3.4.2 NASNet

NASNet is a deep learning architecture where an optimal neural architecture is searched automatically by using the Neural Architecture Search (NAS) method. For the best performance on a specific task, the design of the neural network's topology is automated using the NAS process.

The NAS algorithm can be generalized as an algorithm that searches for the best algorithm to perform a certain task. It involves three different components, namely, the search space, performance estimation strategy, and search strategy. The search space encompasses all the potential architectures that can be looked for within the neural network's subspace. It can be categorized into two primary types: the global search space and the cell-based search space. The global search space offers a high degree of flexibility, accommodating a wide range of architecture due to its ample operation arrangement options. In contrast, the cell-based search space is characterized by recurring fixed structures in effective, manually designed architectures, leading to the assembly of smaller cells into larger architectural structures.

Without construction or training of a possible neural network, the performance is evaluated using the performance estimation strategy, which returns a number or an accuracy value of the possible model architecture, which the NASNet predicts as a possible solution. Different search strategies such as grid search, random search, gradient-based search, evolutionary algorithm, and reinforcement learning can be used to identify the best architectures and avoid bad ones before estimating performance. The steps for NASNet training is shown in Algorithm 6.

3.4.3 Ensemble model

The ensemble consists of five ensemble blocks and a final output block. The input layer receives the output of the VGG-16 fine-tuned model and the NASNet model as a list. This input is then passed through the five ensemble blocks, finally reaching the output layer. Each ensemble block is composed of a fully connected layer, a reshape layer, two convolutional layers, a batch normalization layer, ReLU activation, an ensemble layer, and a max pooling layer.

The ensemble process in the ensemble layer is carried out based on effective moving average. This layer has two parameters, namely,

the decay rate, which is responsible for reducing the effective moving average, and the update rate, which ensures that for every update rate iteration, the weights in the ensemble layer will be modified with the help of the effective moving average.

The output layer is responsible for classification.

The effective moving average is represented mathematically as shown in Equation (3):

$$EMA_{updated} = EMA + (Pred_{NASnet} - Pred_{VGG16FT}) * \delta \quad (3)$$

where $EMA_{updated}$ is the updated effective moving average; EMA denotes the effective moving average before the update operation; $Pred_{NASNet}$ and $Pred_{VGG16FT}$ are the predictions of NASNet and the VGG-16 fine-tuned model, respectively; and δ is the decay rate, which is taken as 0.8 in this case.

The predictions of both models are taken as input. Initially, the prediction of the VGG-16 fine-tuned model was taken as the effective moving average, which is then updated with the help of the above mathematical expression. The update rate ensures that the weights are modified only after a certain number of iterations, which is two in this case. Therefore, for every second iteration, the weights are modified by reshaping the effective moving average tensor for every weight tensor. The reshaped tensor is updated into the weight tensor as the new weight tensor for the next two iterations. The procedure for Ensemble classifier training using EMA is shown in Algorithm 7 and procedure for exponential moving average-based ensemble weight update in a custom ensemble layer is shown in Algorithm 8.

input: *preprocessed tomato leaf image*
output: *trained ensemble with EMA classifier for tomato leaf disease classification*
Function *TrainClassifier(preprocessed_tomato_leaf_image):*
VGG ← *train VGG16 classifier;*
NASNet ← *train NASNet classifier;*
β ← *batch size*
N ← *total classes of tomato leaf diseases*
h ← *height of preprocessed_tomato_leaf_image*
w ← *width of preprocessed_tomato_leaf_image*
c ← *color channels of preprocessed_tomato_leaf_image*
for *epoch* = 1 *to* 100 ***do***
μ ← *learning rate*
while *performance does not plateau* ***do***
batch ← *obtain a batch of size* β
feed ***batch*** *into model through ensemble layer*
$\boldsymbol{output}_{ensemble} \leftarrow EMA\_ensemble(VGG, NASNet)$
feed ***batch*** *into* ***model*** *through fully connected and reshape layers*
$\boldsymbol{output}_{reshape} \cdot \boldsymbol{shape} \leftarrow (h, w, c)$
perform convolution on $output_{reshape}$
$$\boldsymbol{O}(x,y) \leftarrow \sum_{i}^{m_1}\sum_{j}^{m_2}\sum_{k}^{m_c} I(x-i, y-j, k) * K(i,j,k)$$
flatten convolution output
$\boldsymbol{output}_{flatten} \cdot \boldsymbol{shape} \leftarrow (\beta,\ h*w*d)$
feed $output_{flatten}$ *to output layer*
prob ← *predicted tomato leaf disease class probabilities*
labels ← *ground truth probabilities*
loss, δ ← *categorical cross entropy loss*
$$\delta \leftarrow -\log\left(\frac{e^{x_p}}{\sum_{j=1}^{N} e^{x_j}}\right)$$
$x_i \leftarrow$ *logit for class* $i \in \{1, 2, \ldots, N\}$
update model weights θ *using Effective Moving Average,* ***ema***
compute ***accuracy***,
$$accuracy = \frac{\sum_{k=1}^{N}(TP_k + TN_k)}{\sum_{k=1}^{N}(TP_k + TN_k + FP_k + FN_k)}$$
compute ***precision***,
$$precision = \frac{\sum_{k=1}^{n} TP_k}{\sum_{k=1}^{n}(TP_k + FP_k)}$$
compute ***recall***,
$$recall = \frac{\sum_{k=1}^{n} TP_k}{\sum_{k=1}^{n}(TP_k + FN_k)}$$
compute $\boldsymbol{F1-score}$,
$$F1 - Score = \frac{2\sum_{k=1}^{n} TP_k}{\sum_{k=1}^{n}(2TP_k + TN_k + FP_k)}$$
use ***EWG*** *optimizer to monitor loss* δ *and tune model learning;*
end ***while***
if *performance plateaus* ***then***
update learning rate μ *to promote further learning*
end ***if***
end ***for***
$\boldsymbol{output}_{NASNet} \leftarrow$ *output probabilities from model*
return $output_{NASNet}$
end ***Function***
end

Algorithm 7. Ensemble classifier training using EMA for tomato leaf disease classification.

Initialize model ← ***EnsembleClassifier***(*tensor*);
Set ***decay rate***, α ← 0.8;
Set ***update rate***, β ← 2;
Set ***counter*** ← 0;
$\boldsymbol{NasNet}_{Outputs}$ ← *TrainNasNet*(*tensor*);
$\boldsymbol{VGG16}_{Outputs}$ ← *TrainVGG16*(*tensor*);
Function *Custom_EMA_Ensemble*():
Initialize $\boldsymbol{ema}_0$ ← $VGG16_{Outputs}$
while *EnsembleModel is running do*
$\boldsymbol{ema}_i$ ← $(1 - \alpha) * ema_i - 1 + \alpha * NasNet_{Outputs}$
counter ← ***counter*** + 1

```
        if counter % β ← 0 then
        weights
        ←[reshape(ema_i, weight.shape)for weight
            in current model weights]
        Update Ensemble Layer weights
      end if
    end while
  end Function
end
```

Algorithm 8. Exponential moving average-based ensemble weight update in a custom ensemble layer.

3.4.4 Layer information during feature extraction

There are a total of 12 layers used during feature extraction as enumerated below.

3.4.4.1 *(i)* Convolutional layer

The convolutional layer is the most important layer used in CNNs, which is responsible for extracting features from the input with the use of filters or kernels. The kernel is a matrix consisting of a set of learnable parameters. The convolution process can be defined as the conversion of pixels in its receptive field into a single pixel. This operation is performed as the dot product between the kernel matrix and another matrix, which is the receptive field restricted to a certain portion. Hence, in the input image that is composed of three color channels, the kernel carries out the convolution operation in all the three channels, although the height and width will be spatially small. The kernel slides across the height and width of the receptive region of the image. This sliding size is called a stride. The result is a production of a two-dimensional representation of the kernel at each spatial position of the image. The convolution operation results in a feature map as output, which can be represented mathematically as shown in Equation (4):

$$O(x,y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(x-i, y-j) * K(i,j) \quad (4)$$

where $O(x,y)$ represents the value in the output feature map in the position (x, y) and $I(x–I,y–j)$ represents the pixel value in the input at position $(x–i,y–i)$. $K(i,j)$ represents the value of the kernel at position (i,j).

3.4.4.2 *(ii) Depthwise separable convolutional layer*

Depthwise separable convolution handles both the spatial and depth dimensions. Here, the kernels cannot be factored into smaller units. This process is split into two steps:

- Depthwise convolution: a single convolution filter is applied on each input channel.
- Pointwise convolution: it involves the usage of a 1 × 1 filter that iterates through every single point of the input.

This kernel has a depth equal to the number of channels that the input has. The usage of a depthwise separable convolution layer reduces the number of parameters compared to the standard convolution layer.

3.4.4.3 *(iii)* Max pooling layer

The max pooling layer is one of the largely used layers in CNNs, normally found after the convolutional layer. Its purpose is to reduce the spatial dimensions (length and breadth in this case) of the input feature map resulting from the preceding convolution layer. The feature map is taken by the layer as input, which applies the max pooling operation where a window slides through the feature map the window content with the maximum value in the window, thus down-sampling the feature map. Providing a stride value lets the CNN know the number of pixels to move while sliding through that particular layer. The max pooling layer can be mathematically represented as shown in Equation (5):

$$O(x,y) = max_{i=0}^{k-1} max_{j=0}^{k-1} I(x \cdot s + i,\ y \cdot s + j) \quad (5)$$

where $O(x,y)$ is the value in the output feature map at point (x, y), s is the stride value, and $I(x·s+i,y·s+j)$ is the value in the input feature map at position $(x·s+i,y·s+j)$, and k is the size of the pooling window.

3.4.4.4 *(iv)* Average pooling layer

The purpose of using the average pooling layer is to reduce the spatial dimensions such as the length and depth of the feature map just like the max pooling function, but the difference here is that down-sampling is performed by transforming the window into a single value, which is the average of the values present in it. This returns a smoother feature map compared to the max pooling layer, which returns a feature map focusing on prominent features. The average pooling layer can be mathematically represented as shown in Equation (6):

$$Y[i,j,c] = \frac{1}{k_h * k_w} \sum_{p=0}^{k_h-1} \sum_{q=0}^{k_w-1} X[i * s_h + p\ ,\ j * s_w + q, c] \quad (6)$$

where Y is the output after the pooling function, X is the input feature map, k_h is the height of the feature map, and k_w is the width of the feature map. s_w and s_h are the stride values for height and width while sliding through the input feature map.

3.4.4.5 *(v)* Concatenation layer

The concatenation layer concatenates the inputs having the same size in all dimensions except the concatenation dimension, received by the layer along a specified dimension. This layer is used whenever we want to merge the information from different parts of the network or data modalities. The concatenation operation takes place by combining multiple input tensors by stacking them along the specified axis, resulting in a single tensor with an increase in size. The layer is mathematically expressed as shown in Equation (7):

$$O[i,j,c] = \begin{cases} A[i,j,c] & if\ \ 0 \le c < C_1 \\ B[i,j,c-C_1] & if\ \ C_1 \le c \le C_1 + C_2 \end{cases} \quad (7)$$

where O is the output, A is the first input tensor with C_1 channels and B is the second input tensor with C_2 channels for the concatenation layer, i represents the height dimension and ranges from 0 to H, j represents the width dimension and ranges

from j to W, and c represents the channels and ranges from 0 to C_1+C_2.

3.4.4.6 *(vi)* Addition layer

This layer adds inputs from multiple neural network element-wise. This operation can be performed when the input tensors have the same shape. This is done so that the information flows seamlessly through the network just by the addition of the output of one layer to the output of the previous layer. This layer is mathematically represented as shown in Equation (8):

$$O[i,j,c] = A[i,j,c] + B[i,j,c] \tag{8}$$

where O is the output, A is the first input tensor and B is the second input tensor for the addition layer, i represents the height dimension and ranges from 0 to H, j represents the width dimension and ranges from j to W, and c represents the channels and ranges from 0 to C.

3.4.4.7 *(vii)* Batch normalization layer

This layer helps in making neural networks faster and more stable by performing standardization and normalization operations in the feature map that is provided as input to the layer. The normalization process is carried out in two steps:

- Normalization
- Rescaling and offsetting

Before performing normalization, the data are fed into the layer in the form of mini batches. The mean and standard deviations of these mini batches can be found using the following equations shown in Equations (9, 10):

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{9}$$

and

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu)^2 \tag{10}$$

where μ and σ are the mean of the values in the ith value in the mini-batch x of size m.

The main purpose of normalization is to transform the data to have a mean equal to 0 and standard deviation equal to 1, which is carried out using the expression as shown in Equation (11):

$$x_{i(norm)} = \frac{x_i - \mu}{\sigma + \epsilon} \tag{11}$$

Two learnable parameters γ and β are used for rescaling and offsetting, respectively, thereby normalizing each batch accurately. This is represented using the expression shown in Equation (12):

$$x_i = \gamma x_{i(norm)} + \beta \tag{12}$$

where x_i is the ith value of mini batch x and $x_{i(norm)}$ is the normalized ith value of mini batch x.

3.4.4.8 *(viii)* Dropout layer

The dropout layer acts as a mask to nullify some of the neurons' contributions towards the next layer while the rest of the neurons remain unmodified. It aims to prevent overfitting, avoid dependency on a specific neuron during training, and ensure better generalization from the model. The neurons are nullified using a probability for random exclusion such that they behave like they are not part of the architecture. The layer can be mathematically represented as shown in Equations (13, 14):

$$O = X * M \quad during \quad training \tag{13}$$

and

$$O = X * (1-p) during \quad testing \tag{14}$$

where O is the output, X is the input, and p is the probability, and it is scaled to a factor $(1-p)$ during output since the dropout will be turned off during the testing phase. M is a binary mask with the shape same as X and each element of M is set as 0 or 1 depending on p.

3.4.4.9 *(ix)* Global average pooling layer

The global average pooling layer is a pooling layer that performs down-sampling. Unlike the usual pooling layer, the global pooling layer condenses the feature maps into a one-dimensional mapping that can easily be read by the single dense classification layer. The mathematical representation is as shown in Equation (15):

$$O = \frac{1}{H*W} * \sum_{i=0}^{H}\sum_{j=0}^{W}(F[i,\ j]) \tag{15}$$

3.4.4.10 *(x)* Flatten layer

This layer performs the flattening operation that reshapes the input received into a single-dimensional feature vector without affecting the batch. It is done to allow the fully connected layers to operate on the multi-dimensional feature maps since the fully connected layers can only be trained with single-dimensional feature vectors.

3.4.4.11 *(xi)* Fully connected layer

The fully connected layer or simply the dense layer is a CNN layer where all the neurons or nodes in one layer is connected to every node to the next layer. This layer works with activation functions such as the ReLU during feature extraction and softmax during multiclass classification. It is represented as a mathematical function as shown in Equation (16):

$$\mathrm{O} = \mathrm{f}(\mathrm{W}*\mathrm{X} + \mathrm{b}) \tag{16}$$

where X is the input, O is the output, W is the weight matrix, b is the bias vector, and f is the activation layer, which would be ReLU in case of feature extraction and softmax in case of classification.

3.4.4.12 *(xii)* ReLU activation layer

The ReLU is a piecewise linear function used to introduce non-linearity into the feature map obtained as output before the activation function is applied. The ReLU function works by applying a simple thresholding operation where the positive values remain the same while the negative values become zero. The ReLU activation function can be expressed mathematically as shown in Equation (17)

$$f = \max(x, 0) \tag{17}$$

where x is the input given into the function and f is the output obtained.

3.4.5 Classification

3.4.5.1 *(i)* Softmax activation

The softmax activation function is responsible for the multi-class classification of the vector obtained from the convolution layers after the feature extraction phase in the output layer. It works by calculating the exponent of each entry in the vector and dividing the value by the sum of all the exponents in the vector as shown in Equation (18).

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{18}$$

where x is the input vector and i is the ith entry in the input vector with N entries. The denominator of the softmax activation is the sum of the exponents of the entries. This is done for the conversion of N real number entries into a probability distribution of N possible outcomes.

3.4.5.2 *(ii)* Categorical cross-entropy loss function

This loss function (also known as softmax loss) is used with a CNN to provide an output for the probability of each image over N different classes. This function is a combination of softmax activation and the cross-entropy loss function and is thus useful during multi-class classification. Its use allows the comparison of the target and predicted values by the CNN model as an output, thereby measuring the modeling efficiency of the training data by the CNN. The objective of this loss function is to calculate the difference between the ground truth and predicted class distribution. Techniques like gradient descent are used to adjust the weights and biases for minimalization of this loss, thereby improving the predictions. The categorical cross-entropy loss function is written as the negation of logarithmic function of the softmax function as shown in Equation (19):

$$CE = -\log\left(\frac{e^{x_p}}{\sum_{j}^{N} e^{x_j}}\right) \tag{19}$$

where CE is the cross-entropy loss, x_p is the positive class' CNN score, N is the number of classes for classification, and x_j is the jth class' score.

To backpropagate through the network and optimize the defined loss function resulting in tuning the net parameters, the loss' gradient is calculated with respect to the CNN's output neurons given by the gradient of the cross-entropy loss with respect to each CNN's class score. The derivatives are represented mathematically as shown in Equations (20, 21):

Derivative with respect to positive class:

$$\frac{\partial}{\partial x_p}\left(-\log\left(\frac{e^{x_p}}{\sum_{j}^{N} e^{x_j}}\right)\right) = \frac{e^{x_p}}{\sum_{j}^{N} e^{x_j}} - 1 \tag{20}$$

Derivative with respect to negative class:

$$\frac{\partial}{\partial x_n}\left(-\log\left(\frac{e^{x_p}}{\sum_{j}^{N} e^{x_j}}\right)\right) = \frac{e^{x_n}}{\sum_{j}^{N} e^{x_j}} \tag{21}$$

where x_n is the score of any negative class in N other than N_p, which consists of the positive classes.

3.4.6 Optimizer

3.4.6.1 *(i)* Adam optimizer

The Adam optimizer is an extension of the stochastic gradient descent (SGD) algorithm based on adaptive moment estimation, which takes advantage of two principles, namely, the momentum and root mean square propagation (RMSprop). The momentum technique is used to accelerate convergence in gradient descent by adding the fraction of the previous gradient update with the current update, reducing the oscillations. The convergence process speeds up along shadow dimensions, which assists optimization. RMSprop adapts the learning rate for each parameter individually by maintaining a moving average of squared gradients. This helps in scaling learning rates and making the optimization process more robust. With the help of these two methods, the following are obtained as shown in Equations (22, 23):

$$m_t = \beta_1 m_{t_{-1}} + (1 - \beta_1)\left[\frac{\delta L}{\delta W_t}\right] \tag{22}$$

$$v_t = \beta_2 v_{t_{-1}} + (1 - \beta_2)\left[\frac{\delta L}{\delta W_t}\right]^2 \tag{23}$$

where m_t is the estimate of the first-order moment, which is the aggregate of gradients at time t, v_t is the estimate of the second-order moment, which is the sum of the squares of the past gradients at time t, β_1 is the decay rate of average of gradient in the momentum principle, and β_2 is the decay rate of average of gradient in the RMSprop principle. The moment estimates m_t and v_t can be called the weight parameters.

In the Adam optimizer, the bias-corrected weights are considered such that the weight parameters will not be biased towards 0. The bias-corrected weight parameters are as shown in Equations (24, 25):

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t} \tag{24}$$

$$\widehat{v_t} = \frac{v_t}{1-\beta_2^t} \tag{25}$$

These bias-corrected weight parameters are used in the general weight update equation as shown in Equation (26):

$$w_{t+1} = w_t - \widehat{m_t}\left(\frac{\alpha}{\sqrt{\widehat{v_t}}+\epsilon}\right) \tag{26}$$

where α is the learning rate or the step size parameter and ϵ is a small positive constant to avoid division by 0.

3.4.6.2 *(ii)* Enhanced weighted gradient optimizer

This is a modified Adam optimizer that accepts a custom weight as a parameter and incorporates the gradients multiplied by the custom weight into its operation. The custom weights are given as a parameter and are introduced into the gradients with the values being multiplied. The modified values are introduced into the Adam optimizer and then used in our ensemble model. The updated weight with the custom weight parameter before optimization is as shown in Equation (27):

$$\omega = \gamma \cdot w_t \tag{27}$$

This updated weight ω is introduced to the weight update process as shown in Equation (28).

$$w_{t+1} = \omega - \widehat{m_t}\left(\frac{\alpha}{\sqrt{\widehat{v_t}}+\epsilon}\right) \tag{28}$$

where γ is the custom weight parameter, α is the learning rate or the step size parameter, ϵ is a small positive constant to avoid division by 0, w_t is the existing weight before the optimization process, and w_{t+1} is the updated weight after optimization. $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected weight parameters. The procedure for enhanced weighted gradient optimizer is shown in Algorithm 9.

```
Initialize epoch ← 0;
while EnsembleModel is running do
    forward pass
        predictions ← EnsembleModel(batch i);
        loss ← CategoricalCrossEntropy(predictions,
        batch i _labels);
    end forward pass
    backward pass
        gradient, ∇_L ← ∂L/∂θ;
        Custom weights, ∇L custom ←∇L · custom_weight;
        Update EnsembleModel parameters,
        θ ← θ − α · ∇L ;
    end backward pass
    early stopping check
        Monitor validation loss Lval
        Criteria: if Lval does not improve for 4
        consecutive epochs then end training
        if Lval ≤ best loss then
            best loss ← Lval
            patience counter ← 0
        else
            patience counter ← patience counter + 1
            if patience counter ≥ 4 then
                Break training loop;
            end if
        end else
    end early stopping check
end while
end
```

Algorithm 9. Enhanced weighted gradient optimizer.

4 Results and discussion

In this paper, the focus of research starts by addressing a pressing issue in agriculture: the management of plant diseases, with a specific focus on tomato plants. Researchers have employed complex deep learning methodologies and machine learning models to tackle this challenge. This paper strives to revolutionize the ways to identify plant diseases, especially those affecting tomato plants, and manage them accordingly.

The study adopts data analysis and image preprocessing techniques to ensure that the dataset used is well-balanced and that the quality of the images is optimized for deep learning models. It uses methods such as median filtering, resized cropping, and brightness normalization to enhance the features derived from them. This meticulous attention to data quality and balance is crucial in developing a reliable disease classification system. To extract relevant features from the tomato leaf images, the research leverages two transfer learning models, VGG-16 and NASNet. Furthermore, these models are fine-tuned, allowing them to adapt to the specific characteristics of the dataset. This adaptability showcases the potential for pre-trained models to significantly improve classification accuracy when applied to particular datasets.

One of the key novelties is the incorporation of an ensemble model with an EMA function and an EWGO. This innovative approach optimizes the learning process, resulting in a more effective and accurate disease classification system. It stands as a promising method to enhance the performance of machine learning models in agriculture.

4.1 Performance metrics

The evaluation of the models is robust, using a variety of performance metrics, including the confusion matrix, specificity, accuracy, loss, precision, recall, F1-score, ROC curve, AUC, and misclassification rate. These metrics provide a comprehensive assessment of the model's effectiveness, making it clear that the research is backed by rigorous analysis and empirical evidence. The overall proposed architecture is shown in Figure 1, the training data distributions of the dataset is shown in Figure 2, the validated data distributions is shown in Figure 3, images of dataset after preprocessing is shown in Figure 4, images of tomato leaves

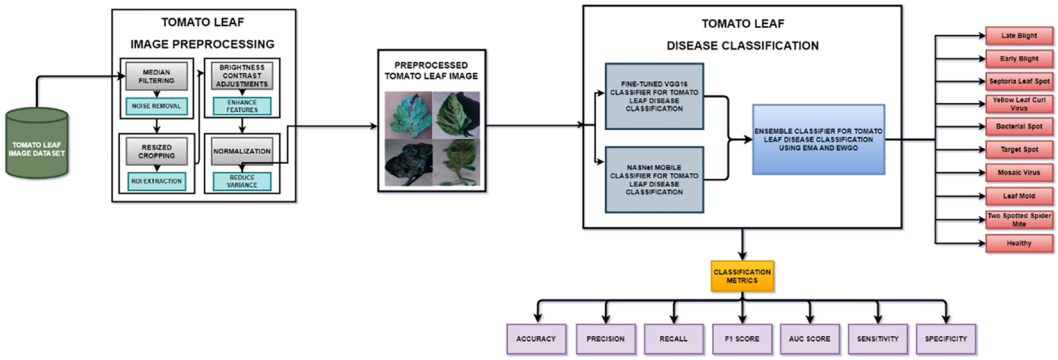

FIGURE 1
Overall proposed architecture for tomato leaf disease classification.

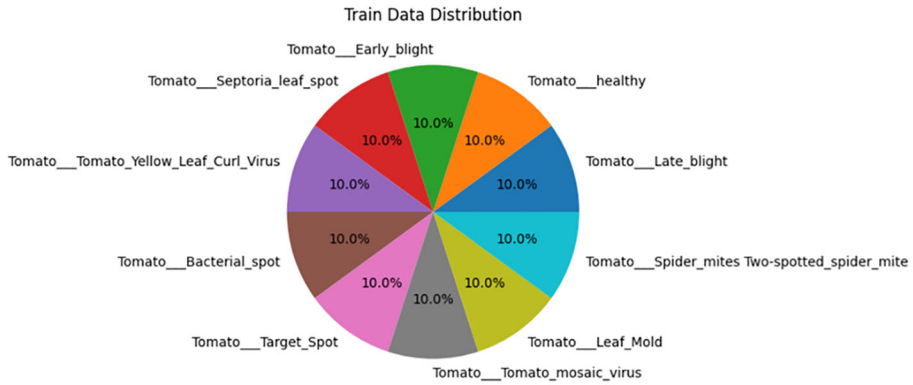

FIGURE 2
Training data distributions of tomato leaf images.

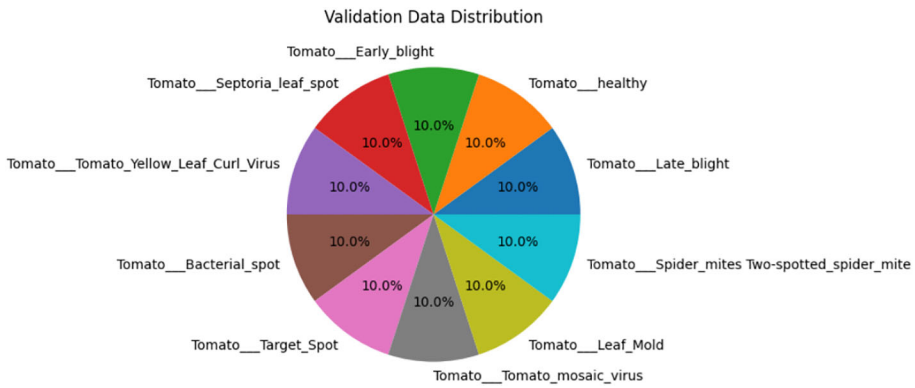

FIGURE 3
Validation data distributions of tomato leaf images.

observed at each preprocessing step in shown in Figure 5, layer architecture for VGG-16 tomato leaf disease classifier is shown in Figure 6, layer architecture for NASNet mobile tomato leaf disease classifier is shown in Figure 7 and layer architecture for ensemble model is shown in Figure 8.

4.1.1 Confusion matrices

The confusion matrix is an $n \times n$ matrix where the rows represent the actual classes while the columns represent the predicted class. The data points are stored in the matrix in cells corresponding to the specific actual class and specific predicted class as count values.

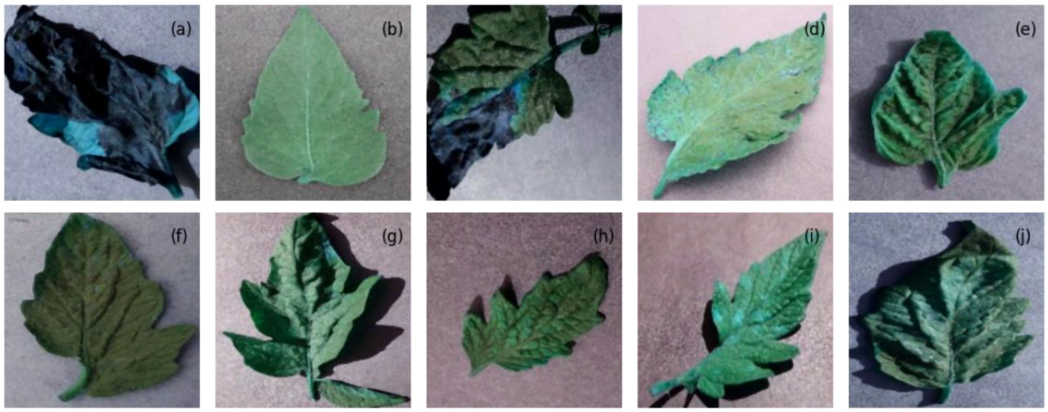

FIGURE 4
The images from the tomato leaf dataset after preprocessing, representing **(A)** late blight, **(B)** healthy, **(C)** early blight, **(D)** septoria leaf spot, **(E)** yellow leaf curl virus, **(F)** bacterial spot, **(G)** target spot, **(H)** mosaic virus, **(I)** leaf mold, and **(J)** two spotted spider mite.

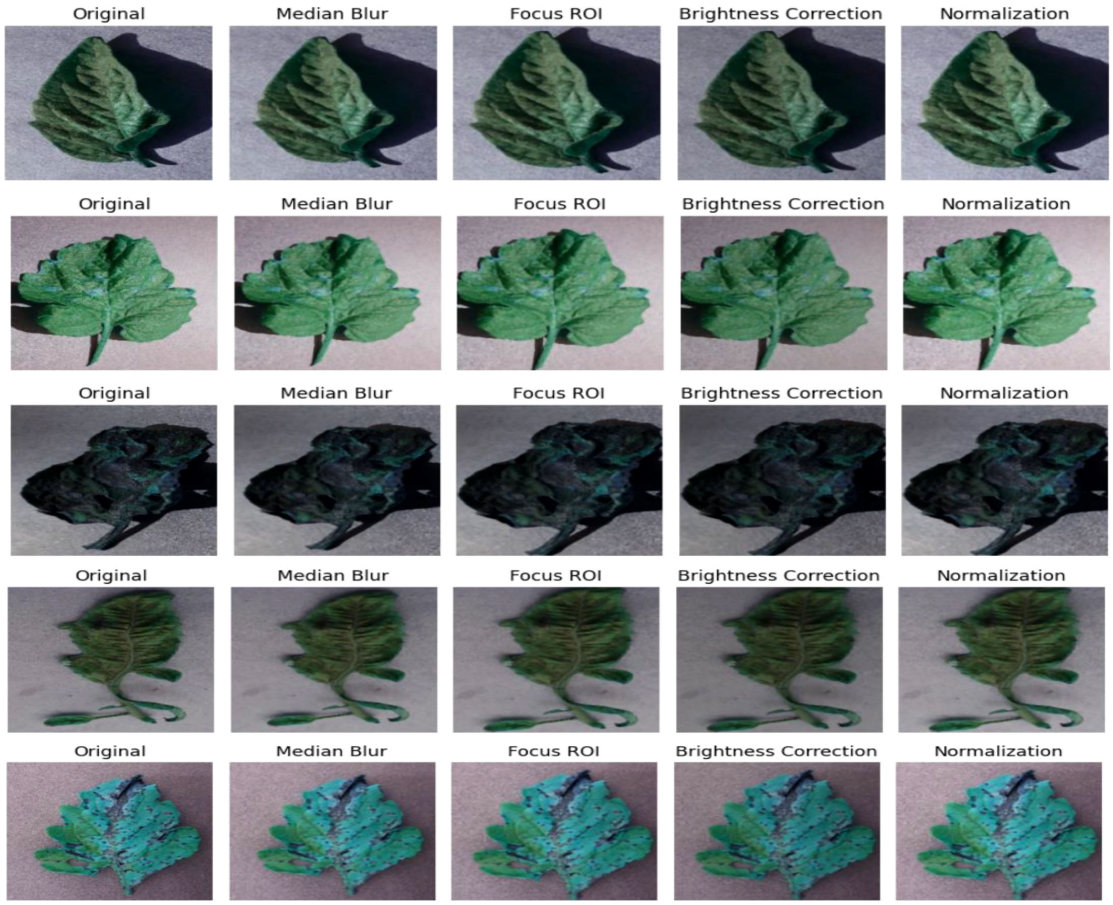

FIGURE 5
Images of tomato leaves observed at each preprocessing step.

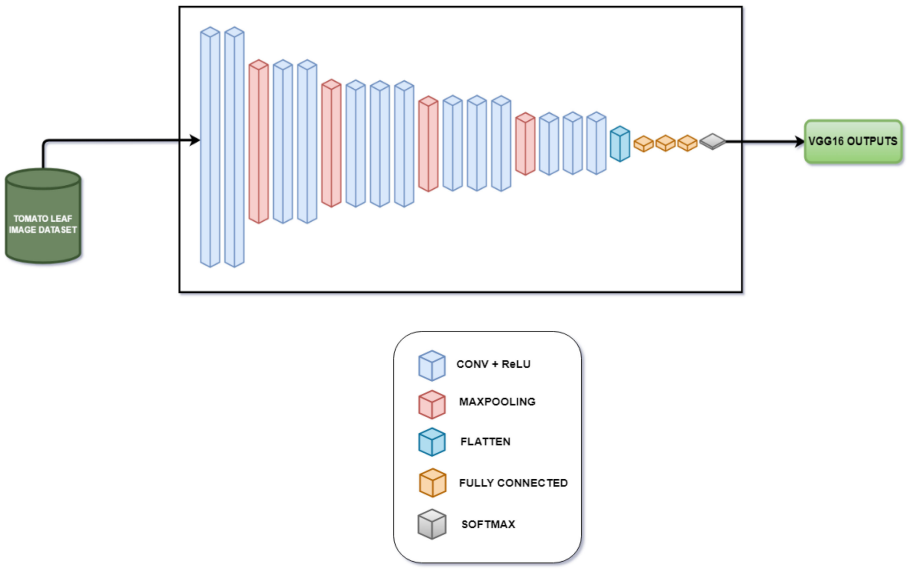

FIGURE 6
Layer architecture for VGG16 tomato leaf disease classifier.

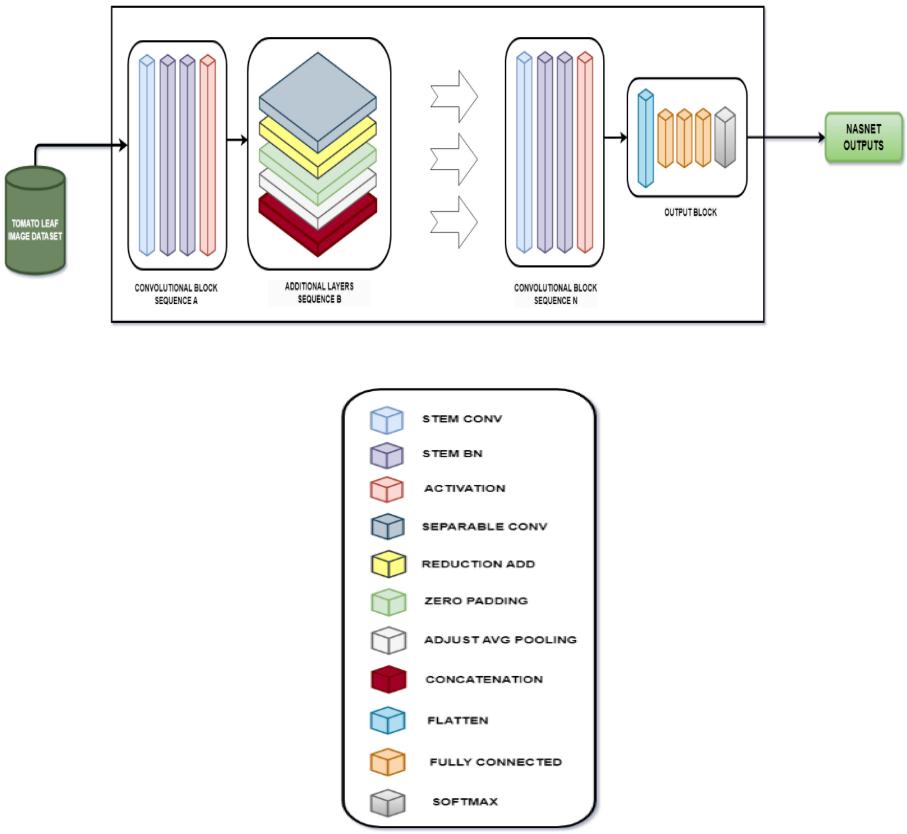

FIGURE 7
Layer architecture for NASNet mobile tomato leaf disease classifier.

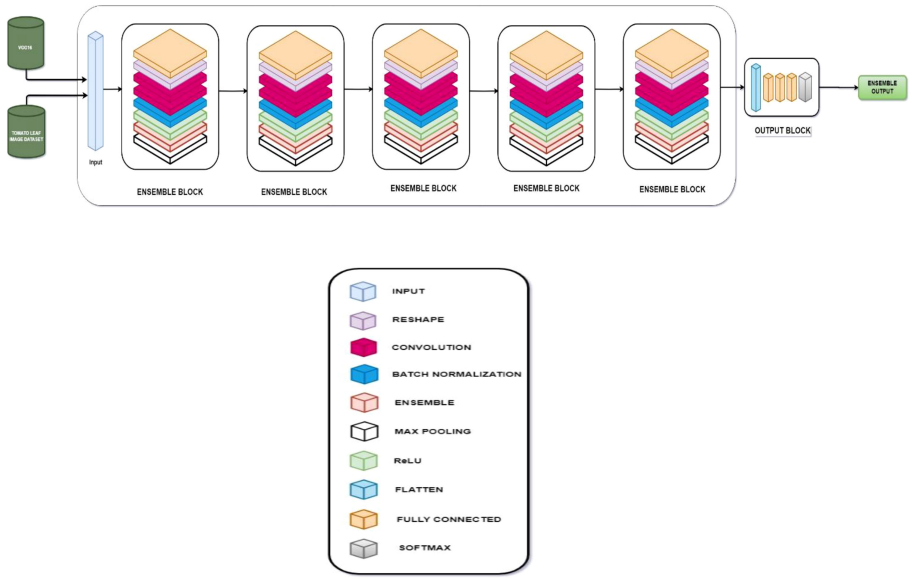

FIGURE 8
Layer architecture for ensemble model.

The above confusion matrix consists of the values predicted by the proposed model corresponding to the actual value. The confusion matrix of the proposed model is shown in Figure 9, the confusion matrix of the VGG-16 fine-tuned model is shown in Supplementary Figure 2, the confusion matrix of the NASNet model is shown in Supplementary Figure 3, the precision values of VGG-16, NASNet, and the proposed model is shown in Supplementary Figure 4.

4.1.2 Specificity

The specificity is the ratio of true negatives to the actual number of negative instances in a specific class. This is a metric to measure the ability of the classifier for correct identification of negative instance within a specific class.

It is mathematically expressed as shown in Equation (29): Specificity of VGG-16, NASNet, and the proposed model is shown in Figure 10.

$$Specificity = \frac{\sum_{k=1}^{n} TN_k}{\sum_{k=1}^{n}(TN_k + FP_k)} \tag{29}$$

4.1.3 Accuracy

Accuracy can be defined as the number of correctly classified images to the total number of images in the dataset. This can be expressed mathematically as shown in Equation (30): Accuracy curves of VGG-16, NASNet, and the proposed model is shown in Figure 11.

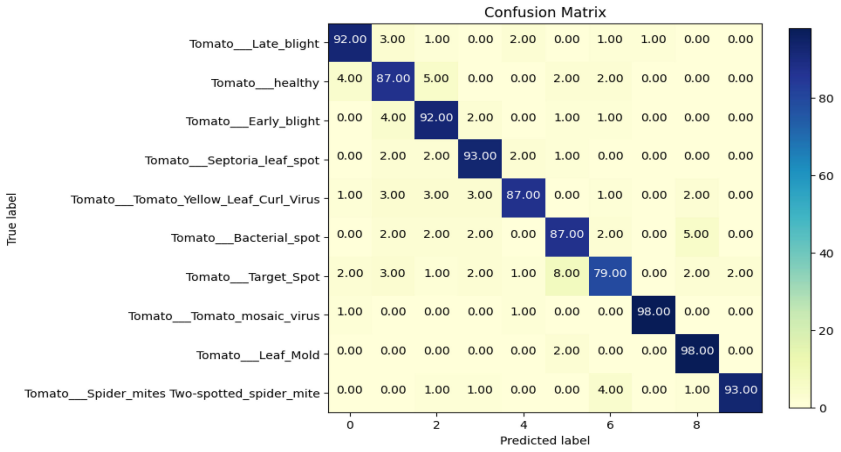

FIGURE 9
Confusion matrix of the proposed model.

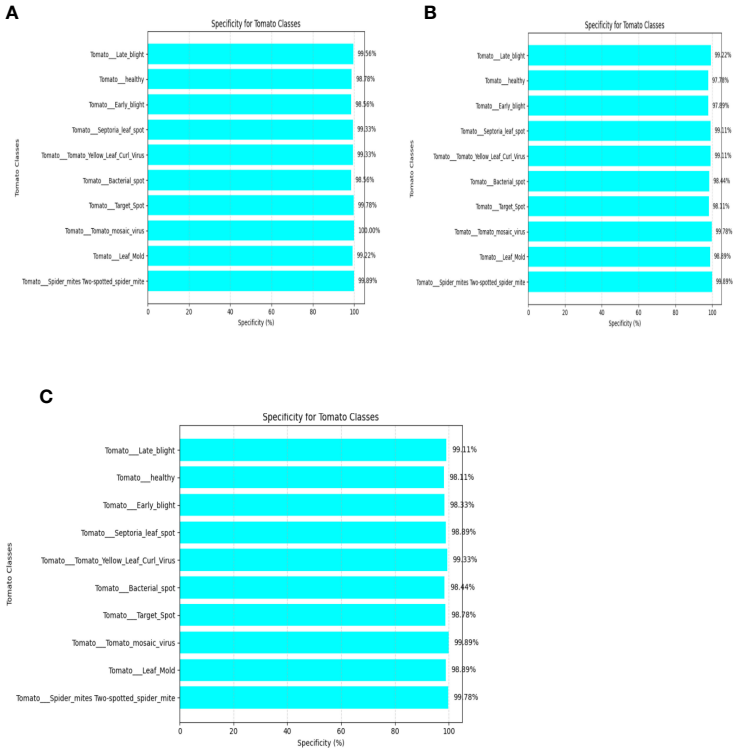

FIGURE 10
Specificity of **(A)** VGG-16, **(B)** NASNet, **(C)** proposed model.

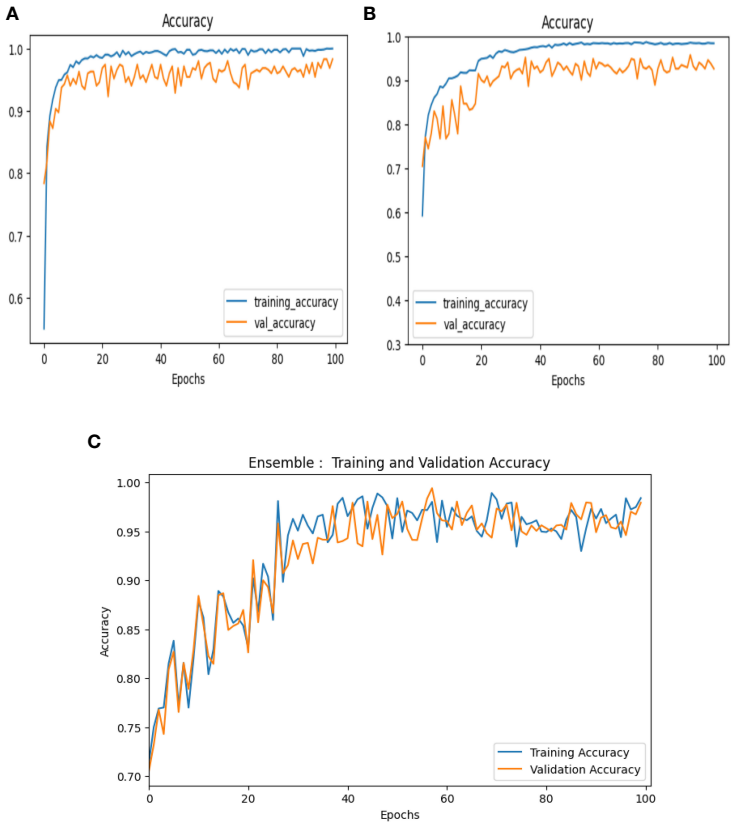

FIGURE 11
Accuracy curves of **(A)** VGG-16, **(B)** NASNet, **(C)** proposed model.

$$Accuracy = \frac{\sum_{k=1}^{n}(TP_k + TN_k)}{\sum_{k=1}^{n}(TP_k + TN_k + FP_k + FN_k)} \quad (30)$$

4.1.4 Loss

Loss is represented as the measure of the model's performance regarding the ability to minimize the difference between the predicted and actual values. In our case, we have used the categorical cross-entropy loss function. Loss Curves of VGG-16, NASNet and the proposed Model are shown in Figure 12.

$$Precision = \frac{\sum_{k=1}^{n} TP_k}{\sum_{k=1}^{n}(TP_k + FP_k)} \quad (31)$$

4.1.5 Precision

Precision is calculated as the ratio of the true total number of instances that are correctly identified as positive by the classifier to the total number of instances identified as positive by the classifier. This is mathematically expressed as shown in Equation (31):

$$Recall = \frac{\sum_{k=1}^{n} TP_k}{\sum_{k=1}^{n}(TP_k + FN_k)} \quad (32)$$

4.1.6 Recall

Recall or sensitivity is the ratio of the number of true positives to the sum of the number of true-positive and false-negative instances in a specific class. This is a metric to measure the ability of the classifier for correct identification of positive instances within a specific class. The recall curves of VGG-16, NASNet, and the proposed model is shown in Figure 13.

It is mathematically expressed as shown in Equation (32):

$$F1\ \ Score = \frac{2\sum_{k=1}^{n} TP_k}{\sum_{k=1}^{n}(2TP_k + TN_k + FP_k)} \quad (33)$$

4.1.7 F1-score

The F1-score is utilized for striking a balance between minimizing the false positives and false negatives and is used as a combination of both precision and recall. Thus, it can be mathematically expressed as shown in Equation (33) and the F1 score curves of VGG-16, NASNet, and the proposed model is shown in Figure 14.

4.1.8 ROC curve and AUC

The receiver operating characteristic (ROC) curve is a graphical representation that consists of the performance of the model in various classification thresholds and is plotted with sensitivity against specificity, thereby visualizing the trade-off between both metrics. AUC helps in quantifying the overall performance of the classifier, which is measured as the area under the ROC curve and the ROC curves of VGG-16, NASNet, and the proposed model is shown in Figure 15.

4.1.9 Misclassification rate

The error rate can be defined as the number of inputs in a particular, which are classified into a wrong class; this can be expressed

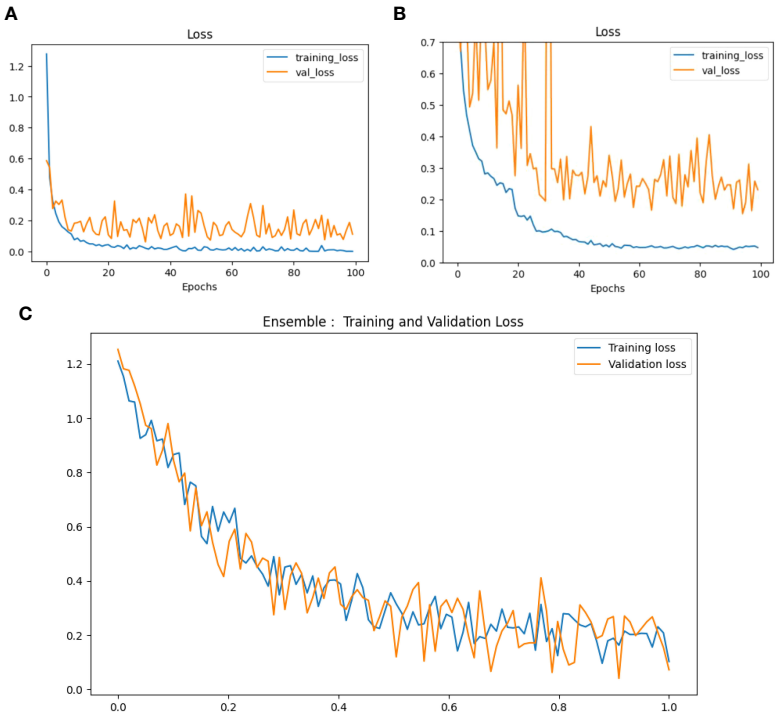

FIGURE 12
Loss curves of **(A)** VGG-16, **(B)** NASNet, **(C)** proposed model.

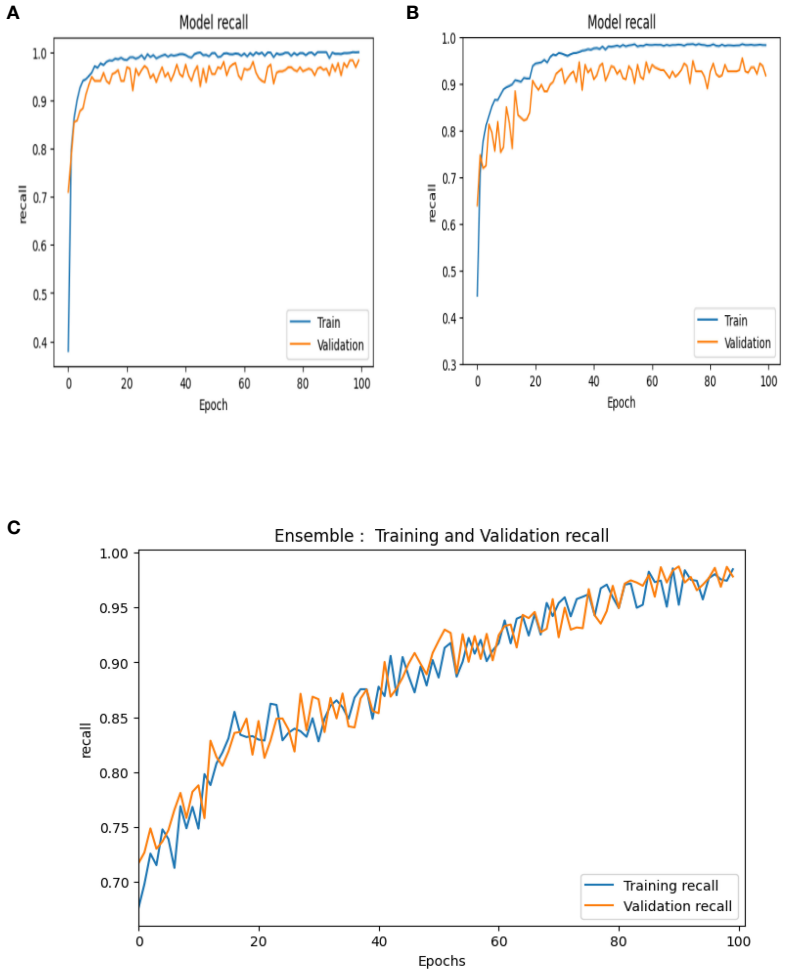

FIGURE 13
Recall curves of **(A)** VGG-16, **(B)** NASNet, **(C)** proposed model.

mathematically as shown in Equation (34): Misclassification rates in VGG-16, NASNet, and proposed model is shown in Figure 16.

$$Error\ \% = \frac{No\ of\ Misclassified\ Instances\ in\ a\ class}{Total\ Number\ of\ Instances\ in\ a\ class} \quad (34)$$

4.2 Performance analysis

The comparison of the three models in the context of the above explained metrics, namely, (a) VGG-16, (b) NASNet, and (c) proposed model, is presented below in graphical representations.

4.3 Interpretation

The above computed performance metrics and the respective graphical representations are proof that the proposed deep learning technique, the suitable application of the ensemble model, and the enhanced classifier and optimizer used have shown a tangible increase of the feasibility in the disease prediction for the given series of input images of tomato leaves. It also proves that the preprocessing procedure applied is a fitting one. The performance values observed for accuracy, loss, precision, recall, ROC, and F1-score are 98.7%,<4%, 97.9%, 98.6%, 99.97%, and 98.7% respectively. It is apparent that the results obtained show significant improvement compared with those shown by conventional and present techniques as explained in the literature. The performance scores recorded for the existing models in the literature are tabulated below. The techniques studied do not record all the performance metrics as in the proposed model in this work. One parameter that is considered in all the models, namely, "accuracy", is exponentially high in the proposed approach. The performance comparison of the proposed model with existing models is shown in Table 1.

4.4 Testing of hypotheses

In order to provide a statistical analysis on the proposed work, testing of hypothesis was carried out in this work. It consists of three hypotheses including a Null hypothesis given in Table 2.

Hypothesis 1: There is a significant influence between season and tomato leaf diseases.

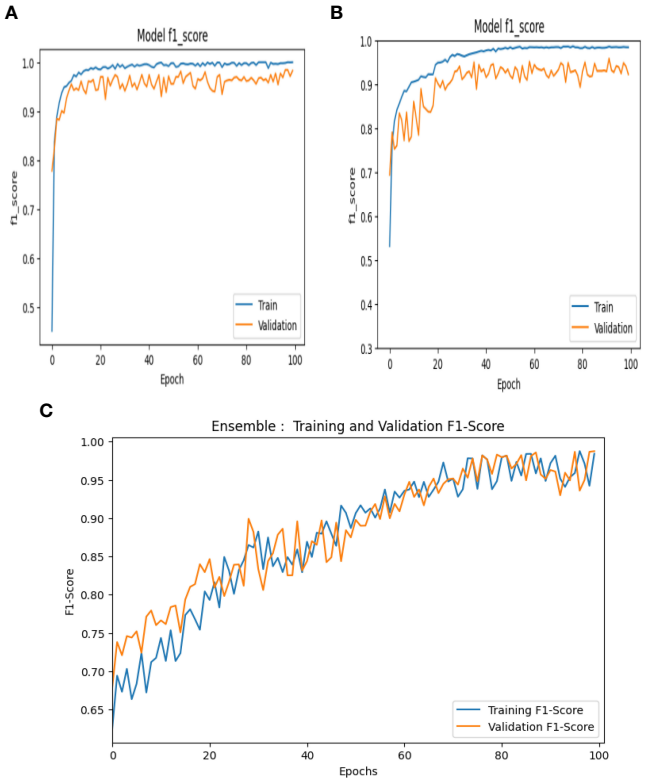

FIGURE 14
F1 score curves of **(A)** VGG-16, **(B)** NASNet, and **(C)** the proposed model.

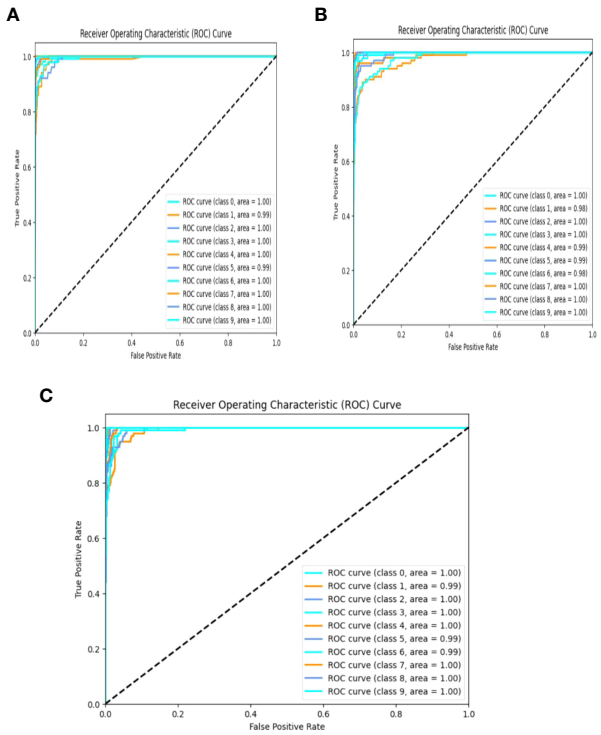

FIGURE 15
ROC curves of **(A)** VGG-16, **(B)** NASNet, and **(C)** the proposed model.

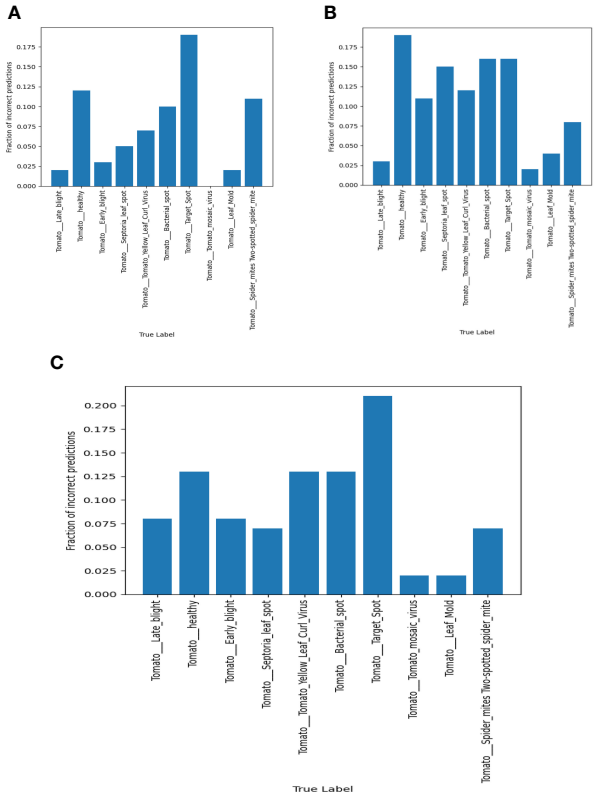

FIGURE 16
Misclassification rates in **(A)** VGG-16, **(B)** NASNet, **(C)** proposed model.

Hypothesis 2: There is no relationship between the occurrence of tomato leaf disease and the environment.

4.5 Testing of Hypothesis 1

As the *p*-value in this test is greater than 0.01, the given null hypothesis can be accepted at the 1% significance level. Hence, there is a significant influence between season and tomato leaf diseases.

Table 3 shows the chi-square test for analyzing the relationship between the deep learning classifier vs. tomato leaf disease detection.

4.6 Testing of Hypothesis 2

H0: There is no relationship between the selection of the deep learning classifier vs. tomato leaf disease detection for performing accurate detection of the disease.

Since the value of *p* is less than 0.5, this hypothesis, which is shown in Table 3, is rejected at the 5% significance level. Therefore, it is concluded that there is a strong and direct relationship between the selection of the deep learning classifier and tomato leaf disease detection from tomato leaf images for performing accurate detection of the disease.

TABLE 1 Performance comparison.

Models	Performance scores (all in %)							
	Specificity	Accuracy	Recall	Precision	F1-score	Loss	ROC	Misclassification
AlexNet (Wang et al., 2017)	–	91.00	91.00	91.0	91.00	-	-	-
GoogLeNet (Wang et al., 2017)	–	94.8	94	94	94	-	-	-
VGG-16 (Wang et al., 2017)	–	95	95	95	95	-	-	-
VGG-16 (Bracino et al., 2020)	–	90.40	-	-	-	-	-	-
LBP M-SVM (Wang et al., 2017)	90.23	97.20	90.75	93.50	-	-	-	-
GPR Quadratic SVM (Ashwinkumar et al., 2022)	–	83.30	-	-	-	-	86.00	-
OMCNN (Khan et al., 2019)	–	98.7	98.2	-	98.5	-	–	-
Proposed adaptive ensemble model	**98.9**	**98.7**	**98.6**	**97.9**	**98.7**	**<4**	**99.97**	**<9**

TABLE 2 H0: There is a significant influence between season and tomato leaf diseases.

Reason for tomato leaf disease	Weighted mean using experiments (observed value *O*)	Weighted mean based on computation (expected value *E*)	$(O - E)^2$	$Value\ is\ \chi^2\ /\ \chi^2 \sum \frac{\leftarrow O\ E \leftarrow^2}{E}$	*p*-value (with 6 dof)
Fungi	7.692	4.649	0.649	4.11	0.65
Fertilizer use	6.329	3.548	0.779		
Bacteria	7.947	4.979	0.612		
Virus	7.309	3.648	0.999		
Viroids	7.519	4.718	0.60		
Geographical Region	6.418	4.269	0.499		

TABLE 3 Analysis of deep learning algorithm's role in tomato leaf disease detection.

Important metric applied on the algorithm	Chi-square value	*p*-value	Mean availability	
			Up to 80%	Above 80%
Accuracy of classification	1.91	0.41	25	11

5 Conclusion and future work

In this research paper, a new ensemble classifier along with an EMA function with temporal constraints, an EWGO that is integrated with two CNN models, namely, VGG-16 and NASNet, has been proposed for the effective detection of diseases in tomato leaves at an early state. This integration of state-of-the-art deep learning CNN technologies with a gradient optimizer and EMA function with temporal constraints provides meticulous data analysis. The proposed model uses image enhancement techniques, and groundbreaking ensemble models underscore a comprehensive approach to tomato leaf disease classification. The amalgamation of image preprocessing, transfer learning, and the pioneering ensemble model with EWGO exhibits promising outcomes in disease classification and increases detection accuracy compared with the existing systems. The main limitation of this work is the lack of time during training. However, an optimizer is added to this work to solve the training time problem. In the future, the implications of this research shall be extended to areas like crop health, global food security, sustainable agriculture, and environmental preservation, underscoring its value within the realm of plant pathology and agriculture.

Data availability statement

The original contributions presented in the study are included in the article/Supplementary Material. Further inquiries can be directed to the corresponding author.

Author contributions

PV: Conceptualization, Methodology, Writing – original draft, Writing – review & editing. AS: Data curation, Investigation, Writing – original draft, Writing – review & editing. JP: Methodology, Supervision, Writing – original draft, Writing – review & editing. SV: Software, Visualization, Writing – original draft, Writing – review & editing. SK: Software, Visualization, Writing – original draft, Writing – review & editing. SA: Data curation, Writing – original draft, Writing – review & editing. AA: Software, Visualization, Writing – original draft, Writing – review & editing. AK: Data curation, Writing – original draft, Writing – review & editing.

Funding

The author(s) declare that no financial support was received for the research, authorship, and/or publication of this article.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fpls.2024.1382416/full#supplementary-material

References

Abed, S. H., Al-Waisy, A. S., Mohammed, H. J., and Al-Fahdawi, S. (2021). A modern deep learning framework in robot vision for automated bean leaves diseases detection. *Int. J. Intell. Robot. Appl.* 5, 235–251. doi: 10.1007/s41315-021-00174-3

Abouelmagd, L. M., Shams, M. Y., Marie, H. S., and Hassanien, A. E. (2024). An optimized capsule neural networks for tomato leaf disease classification. *EURASIP J. Image Video Process.* 2024, 2. doi: 10.1186/s13640-023-00618-9

Agarwal, M., Singh, A., Arjaria, S., Sinha, A., and Gupta, S. (2020). ToLeD: Tomato leaf disease detection using convolution neural network. *Proc. Comput. Sci.* 167, 293–301. doi: 10.1016/j.procs.2020.03.225

Al-gaashani, M. S., Shang, F., Muthanna, M. S., Khayyat, M., and Abd El-Latif, A. A. (2022). Tomato leaf disease classification by exploiting transfer learning and feature concatenation. *IET Image Process.* 16, 913–925. doi: 10.1049/ipr2.12397

Andrushia, A. D., and Patricia, A. T. (2020). Artificial bee colony optimization (ABC) for grape leaves disease detection. *Evol. Syst.* 11, 105–117. doi: 10.1007/s12530-019-09289-2

Anusha, B., Geetha, P., and Kannan, A. (2022). "Parkinson's disease identification in homo sapiens based on hybrid ResNet-SVM and resnet-fuzzy SVM models". *J. Intell. Fuzzy Syst.* 43, 2711–2729. doi: 10.3233/JIFS-220271

Arshad, F., Mateen, M., Hayat, S., Wardah, M., Al-Huda, Z., Gu, Y. H., et al. (2023). PLDPNet: End-to-end hybrid deep learning framework for potato leaf disease prediction. *Alexandria Eng. J.* 78, 406–418. doi: 10.1016/j.aej.2023.07.076

Ashwinkumar, S., Rajagopal, S., Manimaran, V., and Jegajothi, B. (2022). Automated plant leaf disease detection and classification using optimal MobileNet based convolutional neural networks. *Mater. Today: Proc.* 51, 480–487. doi: 10.1016/j.matpr.2021.05.584

Bennet, J., Ganaprakasam, C. A., and Arputharaj, K. (2014). A discrete wavelet-based feature extraction and hybrid classification technique for microarray data analysis. *Sci. World J.* 2014, 1–9. doi: 10.1155/2014/195470

Bhandari, M., Shahi, T. B., Neupane, A., and Walsh, K. B. (2023). Botanicx-ai: Identification of tomato leaf diseases using an explanation-driven deep-learning model. *J. Imag.* 9, 53. doi: 10.3390/jimaging9020053

Bracino, A. A., Concepcion, R. S., Bedruz, R. A. R., Dadios, E. P., and Vicerra, R. R. P. (2020). "Development of a hybrid machine learning model for apple (Malus domestica) health detection and disease classification," in *2020 IEEE 12th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*. 1–6.

Chang, B., Wang, Y., Zhao, X., Li, G., and Yuan, P. (2024). A general-purpose edge-feature guidance module to enhance vision transformers for plant disease identification. *Expert Syst. Appl.* 237, 121638. doi: 10.1016/j.eswa.2023.121638

Chen, H. C., Widodo, A. M., Wisnujati, A., Rahaman, M., Lin, J. C. W., Chen, L., et al. (2022). AlexNet convolutional neural network for disease detection and classification of tomato leaf. *Electronics* 11, 951. doi: 10.3390/electronics11060951

Chouhan, S. S., Singh, U. P., and Jain, S. (2021). Automated plant leaf disease detection and classification using fuzzy based function network. *Wireless Pers. Commun.* 121, 1757–1779. doi: 10.1007/s11277-021-08734-3

Das, S., and Sengupta, S. (2020). "Feature extraction and disease prediction from paddy crops using data mining techniques," in *Computational Intelligence in Pattern Recognition* (Springer), 155–163. doi: 10.1007/978-981-15-2449-3

Debnath, A., Hasan, M. M., Raihan, M., Samrat, N., Alsulami, M. M., Masud, M., et al. (2023). A smartphone-based detection system for tomato leaf disease using efficientNetV2B2 and its explainability with artificial intelligence (AI). *Sensors* 23, 8685. doi: 10.3390/s23218685

Demilie, W. B. (2024). Plant disease detection and classification techniques: a comparative study of the performances. *J. Big Data* 11, 5. doi: 10.1186/s40537-023-00863-9

Dhalia Sweetlin, J., Khanna Nehemiah, H., and Kannan, A. (2016). Patient-specific model based segmentation of lung computed tomographic images. *J. Inf. Sci. Eng.* 32, 1373–1394.

Elgin Christo, V. R., Khanna Nehemiah, H., Minu, B., and Kannan, A. (2019). Correlation-based ensemble feature selection using bioinspired algorithms and classification using backpropagation neural network. *Comput. Math. Methods Med.* 7398307, 1–17. doi: 10.1155/2019/7398307

Elizabeth, D. S., Nehemiah, H. K., Raj, C. S. R., and Kannan, A. (2012). A novel segmentation approach for improving diagnostic accuracy of CAD systems for detecting lung cancer from chest computed tomography images. *J. Data Inf. Qual. (JDIQ)* 3, 1–16. doi: 10.1145/2184442.2184444

Gadade, H. D., and Kirange, D. K. (2022). Deep learning for tomato leaf disease detection for images captured in varying capturing conditions 2191–2200.

Ganapathy, S., Sethukkarasi, R., Yogesh, P., Vijayakumar, P., and Kannan, A. (2014). "An intelligent temporal pattern classification system using fuzzy temporal rules and particle swarm optimization," in *Sadhana*, vol. 39. (Springer), 283–302.

Ganatra, N., and Patel, A. (2020). A multiclass plant leaf disease detection using image processing and machine learning techniques. *Int. J. Emerg. Technol.* 11, 1082–1086.

Han, Z. H., Zhang, Y., Song, C. X., and Zhang, K. S. (2017). Weighted gradient-enhanced kriging for high-dimensional surrogate modelling and design optimization. *Aiaa J.* 55, 4330–4346. doi: 10.2514/1.J055842

Harakannanavara, S. S., Rudagi, J. M., Puranikmath, V. I., Siddiqua, A., and Pramodhini, R. (2022). Plant leaf disease detection using computer vision and machine learning algorithms. *Global Transit. Proc.* 3, 305–310. doi: 10.1016/j.gltp.2022.03.016

Haridasan, A., Thomas, J., and Raj, E. D. (2023). Deep learning system for paddy plant disease detection and classification. *Environ. Monit. Assess.* 195, 120. doi: 10.1007/s10661-022-10656-x

He, J., Liu, T., Li, L., Hu, Y., and Zhou, G. (2023). MFaster r-CNN for maize leaf diseases detection based on machine vision. *Arab. J. Sci. Eng.* 48, 1437–1449. doi: 10.1007/s13369-022-06851-0

Huang, X., Chen, A., Zhou, G., Zhang, X., Wang, J., Peng, N., et al. (2023). Tomato leaf disease detection system based on FC-SNDPN. *Multimed. Tools Appl.* 82, 2121–2144. doi: 10.1007/s11042-021-11790-3

Jabez, C. J., Khanna, N. H., and Arputharaj, K. (2015). "A swarm optimization approach for clinical knowledge mining," in *Computer methods and programs in biomedicine*, vol. 121. (Elsevier), 137–148.

Kaur, P., Harnal, S., Gautam, V., Singh, M. P., and Singh, S. P. (2024). "Performance analysis of segmentation models to detect leaf diseases in tomato plant," in *Multimedia Tools and Applications*, vol. 83. , 16019–16043.

Kaustubh, B. (2020). *Tomato Leaf Disease Dataset* (Kaggle). Available at: https://www.kaggle.com/datasets/kaustubhb999/tomatoleaf?select=tomato.

Khan, M. A., Lali, M. I. U., Sharif, M., Javed, K., Aurangzeb, K., Haider, S. I., et al. (2019). An optimized method for segmentation and classification of apple diseases based on strong correlation and genetic algorithm based feature selection. *IEEE Access* 7, 46261–46277. doi: 10.1109/Access.6287639

Li, X., Li, X., Zhang, S., Zhang, G., Zhang, M., and Shang, H. (2023). SLViT: Shuffle-convolution-based lightweight Vision transformer for effective diagnosis of sugarcane leaf diseases. *J. King Saud University-Comput. Inf. Sci.* 35, 101401. doi: 10.1016/j.jksuci.2022.09.013

Mukhopadhyay, S., Paul, M., Pal, R., and De, D. (2021). Tea leaf disease detection using multi-objective image segmentation. *Multimed. Tools Appl.* 80, 753–771. doi: 10.1007/s11042-020-09567-1

Mustafa, H., Umer, M., Hafeez, U., Hameed, A., Sohaib, A., Ullah, S., et al. (2023). Pepper bell leaf disease detection and classification using optimized convolutional neural network. *Multimed. Tools Appl.* 82, 12065–12080. doi: 10.1007/s11042-022-13737-8

Nahiduzzaman, M., Chowdhury, M. E., Salam, A., Nahid, E., Ahmed, F., Al-Emadi, N., et al. (2023). Explainable deep learning model for automatic mulberry leaf disease classification. *Front. Plant Sci.* 14, 1175515. doi: 10.3389/fpls.2023.1175515

Nerkar, B., and Talbar, S. (2021). Cross-dataset learning for performance improvement of leaf disease detection using reinforced generative adversarial networks. *Int. J. Inf. Technol.* 13, 2305–2312. doi: 10.1007/s41870-021-00772-1

Ngugi, L. C., Abelwahab, M., and Abo-Zahhad, M. (2021). Recent advances in image processing techniques for automated leaf pest and disease recognition–A review. *Inf. Process. Agric.* 8, 27–51. doi: 10.1016/j.inpa.2020.04.004

Nguyen, T.-N., Dong, T. K., Tri, H. X., and Nguyen, C.-N. (2023). "Deep Learning Approach for Tomato Leaf Disease Detection," in *International Conference on Future Data and Security Engineering* (Springer Nature Singapore, Singapore), 572–579.

Pandey, A., and Jain, K. (2022). Plant leaf disease classification using deep attention residual network optimized by opposition-based symbiotic organisms search algorithm. *Neural Comput. Appl.* 34, 21049–21066. doi: 10.1007/s00521-022-07587-6

Pandiyaraju, V., Ganapathy, S., Mohith, N., and Kannan, A. (2023). An optimal energy utilization model for precision agriculture in WSNs using multi-objective clustering and deep learning. *J. King Saud University-Comput. Inf. Sci.* 35, 101803. doi: 10.1016/j.jksuci.2023.101803

Pandiyaraju, V., Logambigai, R., Ganapathy, S., and Kannan, A. (2020). An energy efficient routing algorithm for WSNs using intelligent fuzzy rules in precision agriculture. *Wireless Pers. Commun.* 112, 243–259. doi: 10.1007/s11277-020-07024-8

Pandiyaraju, V., Shunmuga Perumal, P., Kannan, A., and Sai Ramesh, L. (2017). Smart terrace gardening with intelligent roof control algorithm for water conservation 451–455. doi: 10.21162/PAKJAS

Rakesh, S., and Indiramma, M. (2022). "December. Explainable AI for Crop disease detection," in *2022 4th International Conference on Advances in Computing, Communication Control and Networking, ICAC3N*. 1601–1608.

Ruth, J. A., Uma, R., Meenakshi, A., and Ramkumar, P. (2022). Meta-heuristic based deep learning model for leaf diseases detection. *Neural Process. Lett.* 54, 5693–5709. doi: 10.1007/s11063-022-10880-z

Saeed, A., Abdel-Aziz, A. A., Mossad, A., Abdelhamid, M. A., Alkhaled, A. Y., and Mayhoub, M. (2023). Smart detection of tomato leaf diseases using transfer learning-based convolutional neural networks. *Agriculture* 13, 139. doi: 10.3390/agriculture13010139

Sanida, T., Sideris, A., Sanida, M. V., and Dasygenis, M. (2023). Tomato leaf disease identification via two–stage transfer learning approach. *Smart Agric. Technol.* 5, 100275. doi: 10.1016/j.atech.2023.100275

Sankareshwaran, S. P., Jayaraman, G., Muthukumar, P., and Krishnan, A. (2023). Optimizing rice plant disease detection with crossover boosted artificial hummingbird algorithm based AX-RetinaNet. *Environ. Monit. Assess.* 195, 1070. doi: 10.1007/s10661-023-11612-z

Santhosh, D., Pandiyaraju, V., and Kannan, A. (2014). "Non-naive Bayesian classifier for farmer advisory system," in *2014 Sixth International Conference on Advanced Computing (ICoAC)*. 248–254 (IEEE).

Seetharaman, K., and Mahendran, T. (2022). Leaf disease detection in banana plant using gabor extraction and region-based convolution neural network (RCNN). *J. Instit. Eng. (India): Ser. A* 103, 501–507. doi: 10.1007/s40030-022-00628-2

Shoaib, M., Babar, S., Ihsan, U., Ali, F., and Park, S. H. (2022). Deep learning-based segmentation and classification of leaf images for detection of tomato plant disease. *Front. Plant Sci.* 13, 1031748. doi: 10.3389/fpls.2022.1031748

Shoaib, M., Shah, B., Ei-Sappagh, S., Ali, A., Ullah, A., Alenezi, F., et al. (2023). An advanced deep learning models-based plant disease detection: A review of recent research. *Front. Plant Sci.* 14, 1158933. doi: 10.3389/fpls.2023.1158933

Singh, V., and Misra, A. K. (2017). Detection of plant leaf diseases using image segmentation and soft computing techniques. *Inf. Process. Agric.* 4, 41–49. doi: 10.1016/j.inpa.2016.10.005

Sreedevi, A., and Manike, C. (2024). A smart solution for tomato leaf disease classification by modified recurrent neural network with severity computation. *Cybernet. Syst.* 55, 409–449. doi: 10.1080/01969722.2022.2122004

Thai, H. T., Le, K. H., and Nguyen, N. L. T. (2023). FormerLeaf: An efficient vision transformer for Cassava Leaf Disease detection. *Comput. Electron. Agric.* 204, 107518. doi: 10.1016/j.compag.2022.107518

Thangaraj, R., Anandamurugan, S., and Kaliappan, V. K. (2021). Automated tomato leaf disease classification using transfer learning-based deep convolution neural network. *J. Plant Dis. Prot.* 128, 73–86. doi: 10.1007/s41348-020-00403-0

Uma, K., Geetha, P., and Kannan, A. (2016). A novel segmentation of scanned compound images using fuzzy logic. *J. Med. Imaging Health Inf.* 6, 763–768. doi: 10.1166/jmihi.2016.1754

Vallabhajosyula, S., Sistla, V., and Kolli, V. K. K. (2022). Transfer learning-based deep ensemble neural network for plant leaf disease detection. *J. Plant Dis. Prot.* 129, 545–558. doi: 10.1007/s41348-021-00465-8

Wang, G., Sun, Y., and Wang, J. (2017). Automatic image-based plant disease severity estimation using deep learning. *Comput. Intell. Neurosci.*, 2917536. doi: 10.1155/2017/2917536

Wu, K., Li, Q., Chen, Z., Lin, J., Yi, Y., and Chen, M. (2021). Distributed optimization method with weighted gradients for economic dispatch problem of multi-microgrid systems. *Energy* 222, 119898. doi: 10.1016/j.energy.2021.119898

Yakkundimath, R., and Saunshi, G. (2023). Identification of paddy blast disease field images using multi-layer CNN models. *Environ. Monit. Assess.* 195, 646. doi: 10.1007/s10661-023-11252-3

Yu, S., Xie, L., and Huang, Q. (2023). Inception convolutional vision transformers for plant disease identification. *Internet Things* 21, 100650. doi: 10.1016/j.iot.2022.100650

Zhao, Y., Liu, L., Xie, C., Wang, R., Wang, F., Bu, Y., et al. (2020). An effective automatic system deployed in agricultural Internet of Things using Multi-Context Fusion Network towards crop disease recognition in the wild. *Appl. Soft Comput.* 89, 106128. doi: 10.1016/j.asoc.2020.106128