[1]P. Swarajya Lakshmi

[2]Prof. Sanjay Bhargava

Foliage Finder Net based Leaf Localization and NeuroFusionNet based Classification for Tomato Leaf Disease Detection

JES Journal of Electrical Systems

Abstract: - This study presents a thorough framework for the reliable and effective identification of tomato leaf diseases, addressing important issues via a multi-phase procedure. To improve accuracy and dependability, the suggested methodology combines cutting-edge methods for image preprocessing, leaf localization, area of interest (ROI) identification, feature extraction, and a unique neural network-based detection model. First, pre-processed pictures are resized using Lanczos interpolation, normalized to scale pixel values between 0 and 1, and then augmented using rotation, flipping, and zooming. Furthermore, applying median filtering for denoising enhances the quality of the images. Using Focal Loss, the novel FoliageFinderNet—an improved BU-net model—accompanies leaf localization. The ROI identification procedure then makes use of a novel CrayK-Means approach that combines Modified K-means with the Self Adaptive Crayfish Optimization Algorithm (SA-CFOA) for optimum centroid selection. A variety of features, including color histograms, Haralick texture features, circularity-based shape descriptors, and statistical measurements like mean and standard deviation, are used in feature extraction, both manually and deep learning-based. The VGG16 architecture is used to extract features that are based on deep learning. For feature selection, Principal Component Analysis (PCA) is utilized to increase efficiency. NeuroFusionNet, a novel detection method based on Network Fusion with Attention, is introduced in the classification stage. EfficientNet, SqueezeNet, Deep Belief Network (DBN), and Recurrent Neural Network (RNN) versions are combined in this model, which also includes attention mechanisms like Scaled Dot-Product Attention. The all-encompassing strategy is to greatly increase tomato leaf disease detection accuracy and dependability, making a substantial contribution to the control of agricultural diseases.

Keywords: Computing methodologies, Artificial intelligence, Machine learning, Neural networks, Lanczos interpolation, FoliageFinderNet, CrayK-Means approach, NeuroFusionNet

1. Introduction

Tomatoes are a staple of agricultural practises worldwide, but growing them presents considerable hurdles because of their vulnerability to several illnesses. Plant diseases may have a range of effects on tomato crops, from slight symptoms to total plantation destruction, which can result in large financial losses [1]. Given the vital role tomatoes play in both national and global economies, there is a growing need for effective methods to determine and categorise tomato plant diseases so that early intervention can stop extensive harm [2,3]. Conventional disease detection techniques frequently depend on the knowledge of farmers and gardeners, who may find it challenging to control tomato plant growth and precisely identify infections. However, as cutting-edge technology has developed, especially in the fields of deep learning and artificial intelligence (AI), there is a chance to completely change how we tackle the problems associated with disease detection in tomato plants [4,5]. The AI subfield of deep learning has demonstrated impressive results in a variety of applications, from image processing to biomedical image categorization. Deep learning techniques, and more specifically Convolutional Neural Networks (CNN), have become powerful tools for the identification and classification of plant diseases in the agricultural industry, where early disease diagnosis is important to preventing substantial economic losses [6]. The importance of tomatoes to agricultural economies is shown by the fact that Mexico is the world's top supplier, accounting for a sizeable portion of global tomato exports [7].

With the growing demand for tomatoes, it is more important than ever to keep these crops free from disease. It has been determined that early monitoring is essential for selecting the best course of action and stopping the spread of illnesses [8]. But depending just on human observation is error-prone, which emphasises the need for accurate and dependable recognition technology. Deep learning presents a strong answer to problems related to tomato plant disease identification because of its versatility and direct raw data processing capability [9,10]. One subclass of deep artificial neural networks called convolutional neural networks has shown promise in challenging tasks like object recognition and picture classification. These networks are excellent at extracting features from pictures, which makes them suitable for spotting minute alterations in tomato leaves that could be

[1] [1]Research Scholar, Department of Computer Science & Engineering, Mansarovar Global University, Bhopal, India.

[2]Professor, Department of Computer Science & Engineering, Mansarovar Global University, Bhopal, India

signs of illness. The presence of huge and varied datasets for training is a major factor in CNN models' efficacy, even with the potential of deep learning techniques [11,12]. Existing datasets on tomato plant illnesses frequently lack the essential diversity of photos taken under various circumstances, which might cause overfitting and subpar results when tested on actual data.

Researchers use data augmentation methods, including affine and perspective transformations, to improve the dataset and increase the generalizability of the model in order to overcome this issue [13]. In summary, utilising machine learning and deep learning methods especially CNN for the identification and categorization of tomato plant illnesses is a revolutionary strategy for preserving the agricultural sector. The use of cutting-edge technologies highlights the possibility for ongoing innovation at the nexus of artificial intelligence and agriculture, while also enabling more precise and effective disease identification [14,15]. Utilising these technologies becomes essential to maintaining tomato cultivation's resilience and sustainability in the face of changing challenges as long as there is a need for tomatoes worldwide. The following is the paper's primary contribution,

- The proposed FoliageFinderNet, incorporating Focal Loss, presents an innovative approach to leaf localization. This contributes to precise identification and localization of tomato leaves, a critical step in disease detection.
- The study introduces a novel CrayK-Means approach, combining Modified K-means with SA-CFOA, optimizing centroid selection for ROI identification. This innovative technique enhances the accuracy and efficiency of region selection.
- The introduction of NeuroFusionNet, a novel detection model based on Network Fusion with Attention, integrates various neural network architectures (EfficientNet, SqueezeNet, DBN, and RNN) with attention mechanisms like Scaled Dot-Product Attention. This comprehensive fusion enhances disease classification accuracy.

The organization of the paper is as follows, Part 2 gives a summary of current studies on plant disease detection; Part 3 describes the proposed approach; Part 4 contrasts the findings of the proposed framework with those of current techniques; and Part 5 offers a thorough conclusion.

2. Literature Review

The most current publications on tomato disease detection are included in this section.

In 2022, Bhujel *et al.* [16] introduced a lightweight CNN with attention modules, demonstrating its effectiveness in improving model performance while addressing computational constraints. Trained on tomato leaf disease datasets, models incorporating attention mechanisms surpassed standard counterparts by over 1.1%. The convolutional block attention module (CBAM) emerged as particularly effective, contributing to reduced network parameters and complexity.

In 2021, Gadekallu *et al.* [17] utilized a machine learning technique for tomato disease image classification, employing a hybrid approach of principal component analysis and the Whale optimization algorithm. Extracting significant features, the model showcased superiority over classical machine learning techniques, emphasizing accuracy and loss rate metrics in addressing agricultural crises.

In 2020 Wu *et al.* [18] used Generative Adversarial Networks (GANs) to data augmentation to pioneer illness identification. By utilizing deep convolutional GAN in conjunction with the original GoogLeNet inputs, the suggested model was able to get a remarkable top-1 average identification accuracy.

In 2021, Tan *et al.* [19] addressed the pressing challenges in tomato disease detection by comparing the performance of Machine Learning (ML) and Deep Learning (DL) algorithms. Focusing on the PlantVillage dataset, their study filled a critical gap, showcasing the potential of DL in tomato disease classification.

Zhou *et al.* in 2021 [20] used a reorganized residual dense network, which combines the benefits of dense and deep networks, to identify tomato leaf diseases. On the Tomato test dataset, the hybrid model performed better than the others, obtaining a top-1 average identification.

In 2019, Wang *et al.* [21] pioneered the use of deep convolutional neural networks (CNNs) and object detection models, specifically Faster R-CNN and Mask R-CNN, for tomato disease detection. Integrating four deep CNNs with these models, the approach demonstrated accurate identification and segmentation of eleven tomato disease types.

In 2020, Karthik *et al.* [22] focused on automated plant disease detection, utilizing CNNs for feature learning in tomato leaves. Their study presented two architectures, one incorporating residual learning and the other integrating an attention mechanism, achieving an impressive 98% overall accuracy in validation sets during 5-fold cross-validation.

Chen *et al.* in 2020 [23] introduced a novel framework for tomato leaf disease recognition, employing KSW optimized by the Artificial Bee Colony algorithm (ABCK), Binary Wavelet Transform with Retinex (BWTR), and the Both-channel Residual Attention Network (B-ARNet). With 8616 images, the proposed approach achieved an 89% overall detection accuracy, showcasing the efficacy of the ABCK-BWTR and B-ARNet combination.

2.1. Problem Statement

In agriculture, identifying and categorizing tomato plant diseases poses significant hurdles, directly influencing crop yield and global food security. Traditional manual observation methods are susceptible to errors, and the evolving complexity of diseases necessitates more advanced approaches. The integration of deep learning techniques, particularly CNN, holds promise for addressing these challenges. However, the existing literature lacks comprehensive comparisons between ML and DL models for tomato disease detection. This study aims to fill this gap by evaluating and comparing the effectiveness of various ML and DL algorithms. Challenges in this domain include the intricate manifestation of diseases, visual similarities among different ailments, variations in symptom presentation, impact of external factors like environmental conditions and image noise, insufficient and diverse datasets leading to potential overfitting, and the substantial computational resources required for training deep neural networks, especially in resource-constrained environments. Addressing these challenges is essential for developing robust and reliable deep learning models for effective tomato plant disease detection and classification. The complicated ways in which diseases manifest themselves, the visual resemblances of various diseases, the variances in symptom presentation, the impact of outside variables like the surroundings and picture noise, the lack of sufficient and diverse datasets that may lead to overfitting, and the significant computational resources needed for training deep neural networks particularly in environments with limited resources are some of the challenges that this field faces. Solving these problems is necessary in order to build robust and trustworthy deep learning models for effective tomato plant disease detection and classification.

3. Proposed Methodology

To improve accuracy, a systematic approach is used in the proposed method for tomato leaf disease identification. Lanczos interpolation, normalization, augmentation, and the picture pre-processing procedure includes median filtering and denoising. Accurate leaf localization is ensured by FoliageFinderNet, an enhanced BU-net model with Focal Loss. For the best ROI detection, the unique CrayK-Means combines SA-CFOA and Modified K-means. Deep learning features built by hand and based on VGG16 are included in feature extraction. The Principal Component Analysis (PCA) improves the effectiveness of feature selection. For classification, the detection model NeuroFusionNet integrates attention processes with EfficientNet, SqueezeNet, DBN, and RNN. The goal of this simplified approach is to greatly increase the accuracy of tomato leaf disease detection in agricultural settings.

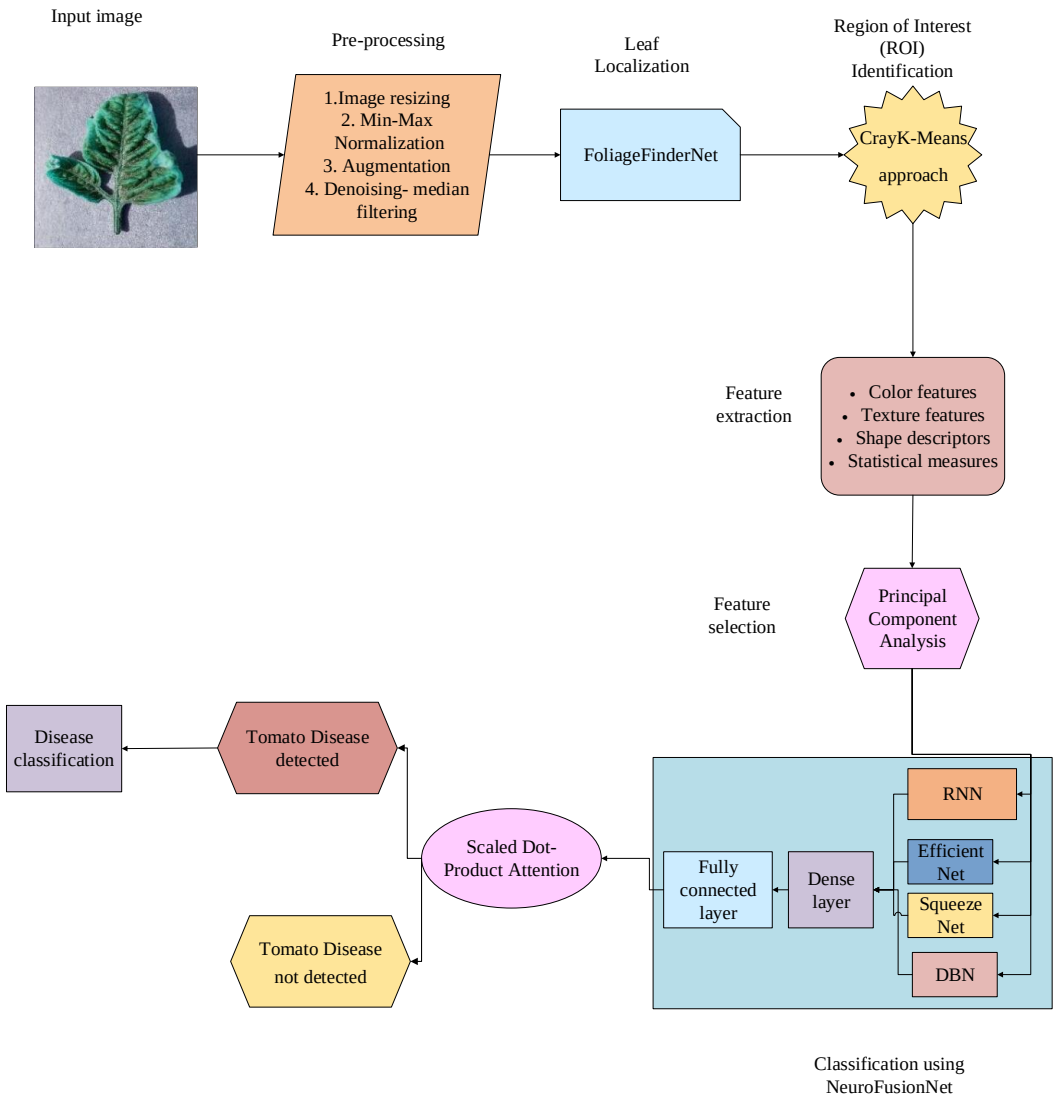


Figure 1: Block diagram of the proposed method for detecting tomato leaf disease

3.1. Pre-processing

Pre-processing involves making necessary adjustments to images in order to increase their quality. This comprises normalization to scale pixel values between 0 and 1, Lanczos interpolation for resizing, and augmentation methods like flipping, rotating, and zooming to provide variety to the collection. In order to lessen picture noise, denoising is also conducted using median filtering.

3.1.1. Image resizing using Lanczos interpolation

A image value can be interpolated between samples using the Lanczos filter. Every sample of the input image is mapped to a scaled and translated copy of the Lanczos kernel. The center hump of a dilated $sinc$ function windows a $sinc$ function, which is known as the Lanczos kernel. A image's sampling rate can be raised by Lanczos resampling. The normalized $sinc$ function $Sinc(x)$, which is windowed by the Lanczos window or $sinc$ window $\left(sinc\left(\frac{x}{a}\right)\right)$ for $-a < x < a$, is known as the Lanczos kernel.

$$L(x) = \begin{cases} Sinc(x)sinc\left(\frac{x}{a}\right), & if\ -a < x < a \\ 0, & Otherwise \end{cases} \quad (1)$$

Equivalently,

$$L(x) = \begin{cases} 1, & if\ x = 0 \\ (\text{asin}(\pi x)\sin(\pi x))/\pi^2 X^2, & if\ 0 < |x| < a \\ 0, & Otherwise \end{cases} \quad (2)$$

Where $a(2,3)$ is a positive integer. It establishes the kernel's size. $(2a - 1)$ lobes, one positive in the center and $(a - 1)$ alternating positive and negative lobe on either side, made up the Lanczos kernel. The discrete convolution of those samples with the Lanczos kernel yields the value $S(x)$ interpolated at a real parameter x for an image with samples S_i.

$$S(x) = \sum_{i=[x]-1+1}^{[x]+a} S_i L(x - i) \quad (3)$$

Where $[x]$ is the floor function and $'a'$ is the parameter for the filter size. Outside of boundaries, the kernel is zero.

3.1.2. Normalization

Normalization is the process that brings the feature's scale to a standard. The classifier's performance and accuracy are enhanced by applying Min-Max normalisation to all returning attributes. The lowest and greatest values for each channel are denoted by x_{min} and x_{max} where X represents the sample data. The objective is to normalize the highest value to 1 and the lowest value to 0, distributing any extra data within this 0–1 range.

$$Norm_{min.max} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (4)$$

3.1.3. Augmentation

Augmentation using methods like rotation, flipping, and zooming.

- **Rotation**

A basic geometric method of data augmentation is rotation. The original photos and the newly produced ones are utilized as training examples after the images are rotated by a predetermined angle.

- **Flipping**

Images can be flipped both vertically and horizontally, or both. This technique is known as flipping. Horizontal flipping, which is more realistic, is the most often utilized flipping technique.

- **Random zooming**

Random zooming is the process of randomly enlarging or contracting the zoom of individual photographs in the collection. This increases the dataset's variability and makes the model more resilient to shifts in perspective and size.

3.1.4. Denoising

In image processing, it is crucial to eliminate noise from the input pictures. It is best to apply the noise reduction stage while preserving as much of the image's clarity as possible and keeping the image's edges unaltered. When using median filtering, a popular low-pass filter, each output pixel is determined by calculating the average brightness values of the surrounding pixels. The size of a pixel in median filtering is calculated by averaging the sizes of its neighbours. The median filter may also be used to easily eliminate the noise caused by salt and pepper. As a result, the median filter is used to pre-process the input images for this investigation. Pixels in median filtering are substituted with the median values of their neighbours.

$$y_{(m,n)} = median(x_{i,j} : (i,j) \in \tau) \quad (5)$$

Where τ denotes the neighbours in (m, n) that are close by.

3.2. Leaf Localization using BU-Net

This step involves localizing leaves using the suggested FoliageFinderNet, a sophisticated model that combines BU-net with Enhanced Leaf Localization by Focal Loss. This novel method uses cutting-edge algorithms to accurately find and identify tomato leaves in images. FoliageFinderNet is a good choice for reliable leaf

localization in the context of tomato leaf disease detection because Focal Loss improves the model's capacity to manage class imbalance.

With a resolution of 256 × 256 for both input and output pictures, the BU-Net architecture is specifically designed for leaf localization. With a decoder on the right and an encoder on the left, it takes on a U-shaped configuration. The output of convolution layers with padding preserves the input dimensions. A max-pooling layer, dropout, and two convolution layers make up an encoder block. Two convolution layers, dropout, concatenation with RES block output, and Conv2DTranspose layers are all involved in decoding. The final decoder block features a convolution layer with six 1 × 1 filters. Images are contracted by the encoder and expanded by the decoder. Facilitating the encoder-to-decoder transition is a broad context block. All convolution layers—aside from the last one, which employs a sigmoid activation—apply batch normalization and ReLU activation. Contraction, expansion, and context-aware blocks are all included into BU-Net's architecture for efficient image processing.

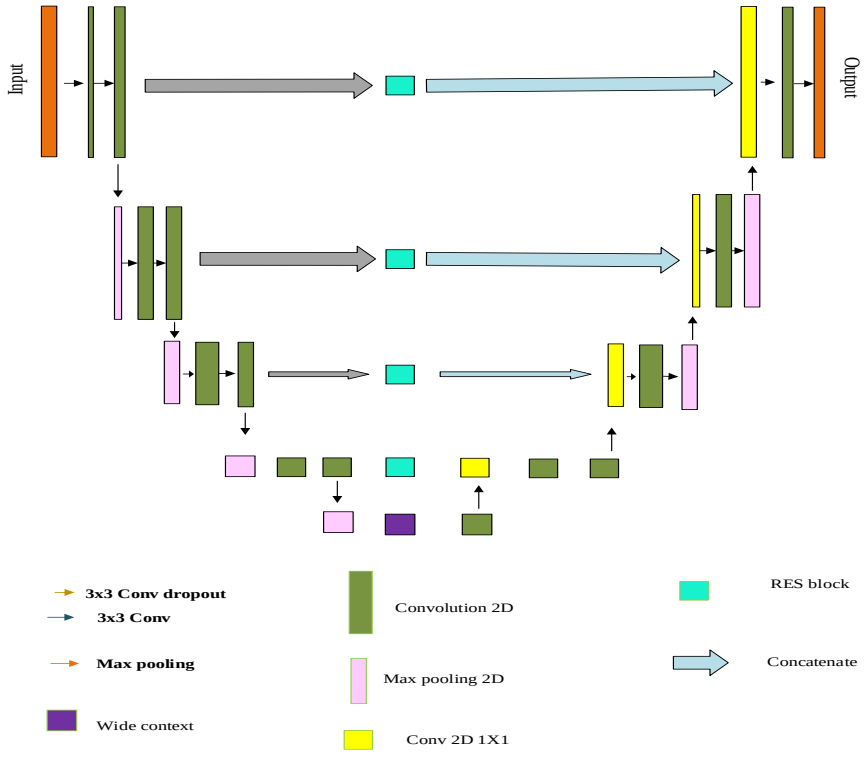
$$ReLU(q) = \begin{cases} 0, & if\ q \leq 0 \\ q, & Otherwise \end{cases} \quad (6)$$

$$Sigmoid(q) = \frac{1}{1+\exp(-q)} \quad (7)$$

Using a process known as hyper-parameter tuning to determine the dropout ratio by testing a variety of values until we found that 0.3 was the best dropout ratio for the network. The customized loss function and Adam optimizer were both employed. Setting the learning rate at 0.01 with a momentum of 0.9 was made. With a patience level of 10, early stopping based on validation loss allowed for the maximum number of training iterations, with a batch size of 16.

3.2.1. Residual Extended Skip (RES)

The RES block's construction is shown in Figure 3. Five parallel connections receive the input distribution. The first four connections apply two convolution layers in a sequential manner, with the first layer using a N × 1 filter size and the second layer using a 1 × N filter size. Because there are fewer parameters overall, the architecture benefits from the choice of cascaded convolution layers. The input is directly forwarded by the fifth connection, which acts as a skip connection. To get a single output, the outputs from the five connections are added together.



The summed output is then subjected to three convolution layers, each with a filter size of 3 × 3, 3 × 3, and 1 x 1. To mitigate information deterioration, the RES block is essential in producing middle-level features from low-level characteristics. It is scale-invariant because it performs contextual aggregation on many scales, which accounts for changes in disease area sizes. The RES block improves segmentation speed in the BU-Net architecture by expanding the valid receptive field. All things considered, the RES block helps with better segmentation by efficiently managing information flow and accounting for size changes in disease zones.

3.2.2. Wide Context (WC)

Both of the parallel connections that have two convolution layers each are created from the input. Both N × 1 and 1 × N filter sizes are used in the first connection, whereas N × 1 and 1 × N filters are used in the second layer of the first connection. The performance is improved overall by this varied filter combination, which adds to a substantial feature set. For the purpose of subclassifying disease types, the WC block gathers contextual information. More accurate reconstruction of segmented areas is made possible by aggregating information at the transition level. A total of the outcomes from both connections is produced as the output. The RES and WC blocks both contribute significantly to enhancing BU-Net's disease picture segmentation skills.

3.2.3. Focal loss function

The widely utilized binary cross-entropy loss is improved into the focused loss by adding a regulating component. For simpler specimens, it reduces the loss of contribution and widens the loss range to low loss. Using the binary cross-entropy loss function, we can define the Focal Loss mathematically using the following equation:

$$L_F(x, GT) = \begin{cases} -\log(x)\sum_{j=1}^{n_0} a(1-x)^b, when\ \ GT = 1 \\ -\log(1-x)\sum_{j=1}^{n_1}(1-a)x^b, Otherwise \end{cases} \quad (8)$$

The modulating factors are $a\ and\ b$, and the pixels in classes 0 and 1 are represented by the numbers n_0 and n_1, respectively. X is the probability value of the system's estimation, and it can have an integer in the range [0, 1]. They can be set in accordance with the requirements and have values in the ranges $(0, 1]\ and\ [0, 5]$ respectively.

3.3. Region of Interest (ROI) Identification via new CrayK-Means approach

The proposed methodology uses a novel CrayK-Means technique to identify the ROI. This new method finds the best centroids by combining the SA-CFOA with K-means. The use of CrayK-Means improves ROI identification accuracy and efficiency, guaranteeing that the most pertinent regions of the pictures are the subject of the ensuing study. This stage is essential for reducing the regions that need to be examined in more depth and improving the overall precision and efficacy of the tomato leaf disease detection procedure.

In the process of clustering, the K-Means method is regarded as one of the most often used algorithms. A data set is classified using this straightforward unsupervised learning approach as a collection of K clusters that are predetermined at the beginning of the process. Its primary goal is to shorten the distance between the cluster head and its members. As can be seen in Figure 1.a, the method starts by selecting an initial K number of clusters. The aim is to generate K clusters from a collection of points xj where 1 ≤ j ≤ N. Using a random selection process, K-Means identifies K points xi with 1 ≤ i ≤ K in the data set as centroids. Each centroid is associated with a cluster C. Next, the closest centroid is assigned to every point in the data set by the method. Based on an objective function, this approach determines the total squared distance for each cluster. Using the objective Eq. (9), the computation is performed:

$$avgmin_c \sum_{i=1}^{K}\sum_{x_j \in C_i} d(x_j, u_i) = avgmin_c \sum_{i=1}^{K}\sum_{x_j \in C_i} \left|x_j - u_i\right|^2 \quad (9)$$

Where the distance between the point and the cluster centroid is expressed as $d(x_j, u_i) = \left|x_j - u_i\right|^2$. The point's location is represented by x_j, while the centroid's position is represented by u_i, where $i = 1, \ldots, K$, where K is the number of clusters. Following the assignment of points to each cluster, the K-Means method employs in Eq. (10) to update the position of each centroid.

$$u_i = \frac{1}{|C_i|}\sum_{j \in C_i} x_j, \forall i \quad (10)$$

Using the Self Adaptive Crayfish Optimization Algorithm (SA-CFOA), to choose centroids as best as possible.

3.3.1. Optimal centroid selection using SA-CFOA

COA draws inspiration from the foraging, summertime vacation, and competitive nature of crayfish. The COA's exploration stage is the summer resort stage, while the exploitation stages are the foraging and competition stages. The algorithm starts with the definition of the crawfish colony X, which stands for the characteristics of swarm intelligence optimization. The position of the i^{th} crayfish, denoted by X_i, suggests a solution, (where dim, also known as dimension, is the characteristic number associated with the optimization issue and $X_i = \{X_{i,1}, X_{i,2}, X_{i,3} \ldots X_{i,dim}\}$). To get a solution X_i introduces the function $f(\cdot)$. Temperature regulates the exploration and exploitation of COA. It is a random constant that represents the surrounding temperature in the region where the subject is located. For COA, the summer resort stage or competition stage will begin when the temperature rises too much. In accordance with the individual location X_i and the cave position X_{shade}, update the new solution throughout the summer resort stage. COA will go into the foraging phase when the temperature is right. When it comes to the foraging stage, the best spot to find food is called the "optimal solution. Food size is determined by the ideal solution, $fitness_{food}$ (obtained by the optimal solution), and the present solution, $fitness_i$ (obtained by X_i). When the food is appropriate, crayfish find new solutions based on where they are. Food position X_{food} update, food intake constant p, and food X_i. When the meal is too big, the crayfish will rip it up with its claw foot before eating in turns with its second and third walking feet. We replicated the crayfish's alternating feeding behaviour using the sine and cosine equations. Crayfish have restricted food consumption. Temperature influences food intake, which has a favourable distribution.

i. Initialize population

Every crayfish in the multiple dimensions optimization issue is a matrix of size 1 × dim. A problem's solution is shown in each column matrix. All variables in a set ($X_{i,1}$, $X_{i,1}$,..., $X_{i,dim}$) must fall within the upper and lower bounds. Randomly creating a collection of potential solutions X in the space is how COA is initially initialized. The population size N and dimension dim serve as the basis for candidate solution X. Eq. (11) illustrates the initialization of the COA algorithm.

$$[X_1, X_2, \ldots, X_N] = \begin{bmatrix} X_{1,1} & \ldots & X_{1,j} & \ldots & X_{1,dim} \\ \vdots & \ldots & \vdots & \ldots & \vdots \\ X_{i,1} & \ldots & X_{i,j} & \ldots & X_{i,dim} \\ \vdots & \ldots & \vdots & \ldots & \vdots \\ X_{N,1} & \ldots & X_{N,j} & \ldots & X_{N,dim} \end{bmatrix} \quad (11)$$

Where $X_{i,j}$ is the location of individual i in the j dimension; N is the number of populations; $X_{i,j}$ value is derived from Eq. (12); and X is the beginning population position.

$$X_{i,j} = lb_j + (ub_j - lb_j) \times rand \quad (12)$$

Where $rand$ is a random integer and lb_j, and ub_j denote the lower and upper bounds of the j^{th} dimension, respectively.

ii. Specify crayfish consumption and temperature

The crayfish will undergo multiple phases of development according to temperature changes. Eq. (13) defines temperature. For their summer vacation, crayfish will select a cool spot when the temperature rises above 30 °C. When the temperature is right, crayfish will start to feed. The temperature has an effect on how much crayfish eat. Crayfish have a feeding range of 15, 30, and 25 °C, which is ideal. As a result, it is possible to compare the crayfish feeding quantity approximately to the typical distribution, suggesting that temperature affects the feeding amount. because between 20 and 30 °C, crayfish exhibit robust foraging behaviour. Accordingly, the COA specifies a temperature range of 20 to 35 °C. Eq. (14) displays the crayfish intake mathematical model.

$$temp = rand \times 15 + 20 \quad (13)$$

Where the temperature in the area where the crayfish is found is represented by the variable $temp$.

$$p = C_1 \times \left(\frac{1}{\sqrt{2\times\pi}\times\sigma} \times \exp\left(-\frac{(temp-\mu)^2}{2\sigma^2}\right)\right) \quad (14)$$

Among them, μ denotes the temperature at which crayfish survive the best, whereas σ and C1 regulate crayfish intake at various temperatures.

iii. Summer resort phase (exploration)

When the temperature rises above thirty, it is too hot. The crayfish will now decide whether to spend their summer vacation within the cave. The following is the definition of the cave X_{shade}:

$$X_{shade} = (X_G + X_L)/2 \quad (15)$$

Where X_L denotes the ideal location of the current population and X_G is the optimal position reached thus far based on the number of repetitions. Searching for caves is an unpredictable process. When $rand < 0.5$, It means that there won't be any competition from other crawfish when the crawfish enters the cave for its summer vacation. At this moment, the crayfish will enter the cave for its summer refuge using Eq. (16).

$$X_{i,j}^{t+1} = X_{i,j}^{t} + C_2 \times rand \times (X_{shade} - X_{i,j}^{t}) \quad (16)$$

According to Eq. (17), C_2 is a declining curve, where t denotes the iteration number of the current generation and $(t+1)$ the iteration number of the following generation.

$$C_2 = 2 - \left(\frac{t}{T}\right) \quad (17)$$

Where T is the maximum number of times that an item can be used. Crayfish wants to get to the cave, which is the best course of action, at the Summer Resort level. This is when the crayfish will come towards the cave. This improves COA's capacity for exploitation while bringing people closer to the ideal answer. Allow a faster convergence of the algorithm.

iv. Competition stage (exploitation)

When $rand$ is less than 0.5 and temp is more than 30, it suggests that the cave also draws in other crayfish. They will now attempt to enter the cave. Using Eq. (18), the crayfish fights over the cave. Time-varying randomness is regulated by the density factor (α), which guarantees a smooth transition from exploration to exploitation. Utilize Eq. (19) to refresh the diminishing factor α, which descends via repetitions to diminish stochasticity gradually:

$$X_{i,j}^{t+1} = X_{i,j}^{t} - X_{z,j}^{t} + X_{shade} + \alpha \quad (18)$$

$$\alpha = C_3 \times \exp\left(\frac{-t}{T}\right) \quad (19)$$

Where C_3 is a constant ≥ 1 (default = 3). Where Eq. (20) indicates that z is the random individual of crayfish.

$$z = round\big(rand \times (N-1)\big) + 1 \quad (20)$$

During the Competition stage, crayfish engage in competition with one another, and crayfish X_i modifies their position in response to another crayfish's position X_z. Position modifications broaden COA's search range and enhance the algorithm's capacity for exploration.

v. Foraging stage (exploitation)

The fish can eat when the temperature is below thirty degrees. The crayfish will now proceed to approach the meal. The crayfish will determine the food's size after finding it. The crayfish will use its claws to tear up any large food, and its second and third walking foot will be used to devour it one at a time. The restaurant $X_{food}$is defined as:

$$X_{food} = X_G \quad (21)$$

The definition of food size Q is:

$$Q = C_4 \times rand \times \left(\frac{fitness_i}{fitness_{food}}\right) \quad (22)$$

Where the values of $fitness_i$, the i^{th} crayfish's fitness value, and $fitness_{food}$, the food location's ftness value, are constant at 3. C_4 is the food factor, signifying the biggest food. The biggest meal is used by the crayfish to determine the appropriate portion size. The meal is considered too large when $Q > (C_4 + 1)/2$. The candy will now be torn by the claw foot of the crayfish. Here is the mathematical formula:

$$X_{food} = \exp\left(-\frac{1}{Q}\right) \times X_{food} \quad (23)$$

As the meal shreds and grows smaller, the second and third paws will alternately pick it up and put it in the mouth. To replicate the alternation process, the sine and cosine functions are mixed. Additionally, since the food that crayfish eat and the food that they consume are linked, the following is the formula for foraging:

$$X_{i,j}^{t+1} = X_{i,j}^{t} + X_{food} \times p \times (\cos(2 \times \pi \times rand) - \sin(2 \times \pi \times rand)) \quad (24)$$

The following equation determines when $Q < (C_4 + 1)/2$, at which point the crayfish must approach the meal and begin eating:

$$X_{i,j}^{t+1} = \left(X_{i,j}^{t} - X_{food}\right) \times p + p \times rand \times X_{i,j}^{t} \quad (25)$$

At each step of the foraging process, crayfish utilize different eating strategies based on the size of their meal Q; food X_{food} is the best option. Once it has reached a size suitable for its consumption, the crayfish will come towards the food. If Q is too high, it suggests a significant discrepancy between the best possible solution and the chaotic one. Consequently, X_{food} requires to be lowered and moved nearer to the meal and control the unpredictable nature of the Crayfish food intake augmentation mechanism. The algorithm's exploitation and convergence skills will be enhanced as COA approaches the optimal solution during the foraging stage.

3.4. Feature Extraction

Tomato leaf disease detection accuracy is achieved by the analysis of several features of the areas that have been discovered during the feature extraction step. Statistical metrics such as mean and standard deviation, circularity-based descriptors, color histograms, and Haralick texture characteristics are examples of handcrafted features. The VGG16 architecture is used to extract features based on deep learning simultaneously. By offering a thorough depiction of visual characteristics, this dual method establishes a vital basis for later phases and raises the detection process's overall accuracy.

3.4.1. Handcrafted Features

- **Colour features-Colour histograms**

The ratio of various hues across the image is represented by the colour histogram, which is independent of the spatial location at any given place. The colour histogram is defined as follows.

$$H = \{h[C_1], h[C_{12}], \ldots., h[C_k] = 1, 0 \leq h[C_k] \leq 1\} \quad (26)$$

The following shows the pixel frequency ($h[C_k]$) of the Kth colour that appears in the picture,

$$h[C_k] = \frac{\sum_{i=0}^{N_1-1} \sum_{j=0}^{N_2-1} \begin{cases} 1(I(i,j)=c_k \\ 0, \ Other \end{cases}}{N_1 \times N_2} \quad (27)$$

Where N_1 and N_2 stand for the image's width and height, respectively.

- **Texture features - Haralick features**

Haralick characteristics represent the brightness relationship between neighbouring pixels quite well. Based on a co-occurrence matrix, Haralick provided fourteen measures for textual attributes. It illustrates the relationship between the intensity of a certain pixel at a given location and the nearby pixels. The Gray-Level Cooccurrence Matrix (GLCM) is developed for images of size N, where N is the number of intensity levels in the image. GLCM is developed to evaluate fourteen characteristics for every picture. Haralick used the normalized GLCM matrix, C_n, to produce 24 unique statistical features. These characteristics significantly contribute to the

quantification of the image's spatial and local information. Only five of the 24 qualities are highly promising and are thus included in this study's Eq. (28) to Eq. (32) after a few experimental combinations were constructed to obtain the necessary data, even though all 24 attributes may be significant in different textural investigations.

Homogeneity: The degree of closeness between each GLCM element and its diagonal elements is known as homogeneity.

$$Homogenity = \sum_{m=1}^{M}\sum_{n=1}^{N}\frac{C_n(i,j)}{1+(i-j)^2} \quad (28)$$

Entropy: A measure of disorderliness or unpredictability in an image is called entropy.

$$Entropy = -\sum_{m=1}^{M}\sum_{n=1}^{N} C_n(i,j)\ln C_n(i,j) \quad (29)$$

Energy: The energy that gives measurements of the geographical homogeneity of the gray levels is the source of the angular second moment.

$$Energy = \sqrt{\sum_{m=1}^{M}\sum_{n=1}^{N} C_n^2(i,j)} \quad (30)$$

Correlation: The feature of the cooccurrence matrix that shows the graylevel values' linear dependency.

$$Correlation = \sum_{m=1}^{M}\sum_{n=1}^{N} C_n(i,j)\frac{(i-\mu_x)(j-\mu_y)}{\sigma_x\sigma_y} \quad (31)$$

Where the long horizontal and vertical spatial planes, , respectively, denote the means and standard deviations of the summed cooccurrence matrix C_n. The measure of gray-level intensity fluctuation between pixels is known as contrast, sometimes known as standard deviation.

The long horizontal and vertical spatial planes, designated as μ_x, μ_y, and σ_x, σ_y, respectively, represent the means and standard deviations of the summed cooccurrence matrix C_n. Standard deviation, commonly referred to as contrast, is a measurement of the variation in grayscale intensity between pixels.

$$C = \sum_{m=1}^{M}\sum_{n=1}^{N}(i-j)^2\, C_n(i,j) \quad (32)$$

- **Shape descriptors – circularity**

The degree to which a form resembles a circle is called its circularity. It is described as the relationship between a shape's area and the size of the smallest enclosing circle. The circularity of a circle is 1, whereas the circularities of other forms are smaller. The circularity may be computed using the following formula:

$$\mathrm{C} = 4\pi\mathrm{A}/\mathrm{P}^2 \quad (33)$$

Where C is the circularity, A is the area of the shape, and P is the perimeter of the shape. Circularity is often used to distinguish between circular and elongated objects.

- **Statistical measures - mean, standard deviation for each region**

i. Mean

The mean, commonly referred to as the average, is a statistic that depicts the arithmetic mean of a group of data and serves as a gauge of central tendency. By averaging all of the dataset's variables and dividing the result by the total amount of variables, it is computed.

$$Mean = \frac{Sum\ of\ all\ values}{Total\ number\ of\ values} \quad (34)$$

ii. Standard Deviation (SD)

The measurement of the data's dispersion from the mean is referred to as the standard deviation (σ). A low standard deviation indicates that the data are concentrated around the mean, while a big standard deviation indicates a greater dispersion of the data.

$$SD\ (\sigma) = \sqrt{\frac{\sum(x_i-\mu)^2}{N}} \quad (35)$$

Where x_i is the input variable, μ is the mean value and N is the total number of input variables.

3.4.2. Deep Learning-based Features using VGG 16

The bottleneck features were extracted using the VGG16 deep neural network, which was pre-trained on the ImageNet dataset. Three completely linked layers are included in the design, which consists of five initial blocks of convolutional layers. A 3x3 kernel with a stride of 1 and padding of 1 preserves spatial dimensions in every convolutional layer. The transferred convolutional layer configurations are used to derive bottleneck characteristics. Spatial dimension reduction is accomplished using max pooling, which uses a 2x2 kernel with a stride of 2 and no padding after each convolution, and ReLU activation. Two completely linked layers with 4,096 ReLU activation units each lead to a final 1,000-unit softmax layer as the architecture comes to a close. The VGG16 model has significant evaluation costs, high memory needs, and a large number of parameters—more than 138 million—despite its effectiveness. Approximately 123 million of these parameters are found in the fully-connected layers.

3.5. Feature Selection using PCA

To maximize and enhance the efficacy of the feature set, PCA is utilized for feature selection. PCA reduces dimensionality while locating and keeping the most useful features. A linear dimensionality reduction method known as PCA involves reducing the dimensionality of data that is very large by the lowest dimension's variance should be maximized. The computation of the feature vector's covariance matrix comes first, and its associated eigenvectors are calculated after that. It must implement feature scaling or normalization similar to supervised learning approaches depending on the n-dimensional training data $y^{(1)}, y^{(2)}, \ldots, y^{(n)}$. Eq. (36) is used to compute the mean of each characteristic.

$$\mu_i = \frac{1}{n}\sum_{j=1}^{n} y_i^{(j)} \quad (36)$$

That can scale separate characteristics so that they have comparable ranges of mean values if they possess distinct mean values for different attributes. To guarantee that every y_i value has an absolute zero mean value, then swap out each y_i variable with its corresponding $y_i - \mu_i$ value. This assessment of the i^{th} element in the framework of supervised learning is described by Eq. (37), while s_i is the |max-mean| value of the i^{th} characteristic.

$$y_i^{(j)} = \frac{y_i^{(j)} - \mu_i}{s_i} \quad (37)$$

In order to decrease the feature's degree from N to m and identify the area in an area with N dimensions into where the data being projected are situated, it is essential to calculate the average squared error of the data that is displayed on the m multivariate vector. It is difficult and outside the scope of this research to computationally verify the anticipated points: $wu^{(1)}, w^{(2)}, \ldots, w^{(N)}$ on these vectors and the computation of these m vectors $\mu^{(1)}, \mu^{(2)}, \ldots, \mu^{(m)}$. Eq. (38) is used to calculate the covariance matrix, which has the dimensions $N \times 1$ for the $y^{(j)}$ vector and $1 \times N$ for the $(y^{(j)})^T$. This results in a covariance matrix with the dimensions $N \times N$. The new magnitude and corresponding orientations of the feature vectors in the changed vector space are represented by the covariance matrix's eigenvalues and eigenvectors, which are then determined. Eigenvectors with small eigenvalues, on the other hand, carry very less information about the dataset. The covariance matrix is given in Eq. (38),

$$C_v = \frac{1}{N}\sum_{j=1}^{N} y^{(j)} \times (y^{(j)})^T \quad (38)$$

Where C_v is the covariance matrix. As a consequence, the formula $t^{(p)} = y^{(j)} z^{(p)}$ may be used to calculate the score for the p^{th} entire PCA of a data vector, where $z^{(p)}$ is the p^{th} eigen value of the C_v of $y^{(j)}$. As a result, the entire PCA reduction of the vector Y may be illustrated as $T = Y \times Z$, where Z is the eigen values of the C_v. Then we establish a cutoff value at which the eigenvalues are deemed valuable and the other ones are eliminated as irrelevant characteristics.

3.6. Disease classification using NeuroFusionNet

In disease classification, the NeuroFusionNet model combines variants like RNN, EfficientNet, SqueezeNet, and DBN. The integration of attention mechanisms, including Scaled Dot-Product Attention, enhances the

model's ability to capture intricate patterns. This fusion of diverse architectures ensures a comprehensive analysis of spatial and temporal features, contributing to robust disease classification. The attention mechanisms refine the model's focus on crucial elements, resulting in an accurate and reliable categorization of tomato leaf diseases, concluding the proposed methodology with a sophisticated disease classification approach.

3.6.1. RNN

Time series problems are solved with a dynamic neural network known as an RNN. Unlike MLPs, RNNs utilize connections between nearby hidden neurons. Time-dependent input data in the sliding window may be gradually relayed through hidden units, allowing temporal correlations between distant events in the time dimension to be considered through these linkages. An RNN's hidden unit structure in general. The basic equations related to the structure of a typical RNN are displayed below. Eq. (39) displays the hidden unit's output for the typical RNN at time t. The mean squared error (MSE) is the loss function at time t, and it is represented by Eq. (40). The derivative of the loss L_t with respect to the weights is shown in Eq. (41).

$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b) \quad (39)$$

$$L_t = \frac{1}{2}(u_t - p_t)^2 \quad (40)$$

$$\frac{\partial L_t}{\partial W_x} = \sum_{k=0}^{t} \frac{\partial L_t}{\partial p_t}\frac{\partial p_t}{\partial h_t}\frac{\partial h_t}{\partial h_k}\frac{\partial h_k}{\partial W_x} \quad (41)$$

$$\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} = \prod_{j=k+1}^{t} diag\left(tanh'(h_j)\right) W_h \quad (42)$$

Where the output vectors of hidden units at times $t, j, and\ k$ are respectively, h_t, h_k, and x_t, the input vector at time t; The hidden layer's input and connected weight matrices for the output are denoted by the letters W_x and W_h; The term with bias is b. The hyperbolic tangent activation function is represented by $tanh()$, which is $\tanh(x) = \frac{1-e^{2x}}{1+e^{2x}}$. On the main diagonal, $diag()$ yields a diagonal matrix containing the vector elements.

3.6.2. DBN

DBNs replicate the underlying data representation through effective layer-by-layer supervised learning. An RBM is a kind of neural network that serves as the basis for a DBN. A two-layer structure having one visible layer and one concealed layer is called an RBM. The visible and hidden layers have been connected using symmetric weights rather than directed ones, and the units inside a layer are still not linked in an RBM. The hidden units construct the high information links observed at the visible units. The generating weights of an RBM are solved using a greedy layer-wise pretraining technique. The data pre-processing algorithm is an unsupervised algorithm that defines the generating weights of an RBM using unlabelled data. The weights of an RBM for the visible and hidden $E\ (v, h)$ define the energy of a joint configuration, which is described as,

$$En(v, h; \theta) = -\sum_{n=1}^{V}\sum_{m=1}^{H} v_s h_t W_{st} - \sum_{n=1}^{V} b_n v_s - \sum_{m=1}^{H} a_m h_s \quad (43)$$

Where θ = {W, b, a}, v_s, h_t binary state of visible s and hidden t, ($v_s, h_t\ \epsilon$ {0,1}). Then, W stands for the symmetric weight parameters with dimensions of $V \times H$, where V is the number of visible units, H is the number of hidden units, a and b are the bias parameters of visible and hidden units, respectively.

As shown in Eq. (44), the energy function is utilized to determine a probability for every potential visible-hidden vector pair in an RBM.

$$p(v, h) \propto \exp\left(-E(v, h)\right) \quad (44)$$

The derivative of log probability of a visible vector with regard to the weights is identified by Eq. (45).

$$\frac{\partial \log P(v)}{\partial w_{ji}} = \langle v_a, h_b \rangle_{data} - \langle v_a, h_b \rangle_{model} \quad (45)$$

The angle bracket denotes the expectation with respect to the specified distribution in the subscript. The training data expectation is $\langle v_a, h_b \rangle_{data}$, and the model expectation is $\langle v_a, h_b \rangle_{model}$. The update rule for the weights follows the gradient of the log as represented in Eqn. (46), where ε is the learning rate.

$$\Delta w_{mn} = \varepsilon(\langle v_a, h_b \rangle_{data} - \langle v_a, h_b \rangle_{model}) \quad (46)$$

To estimate Δw_{mn}, contrastive divergence learning is employed. Eq. (48) is used to determine the apparent units of binary states, whereas Eq. (47) is used to calculate the hidden units. The computed visible states are a reconstruction of the original observable vector. The learning rule offers an adequate and feasible way to estimate the training set's gradient, enabling the discovery of the best features.

$$p(h_t = 1) = \frac{1}{1+\exp\left(-u_t - \sum_s v_s w_{mn}\right)} \quad (47)$$

$$p(v_s = 1) = \frac{1}{1+\exp\left(-x_s - \sum_t h_t w_{nm}\right)} \quad (48)$$

It is demonstrated that just one step is required to approximate optimal acquisition, leading to a large decrease in training time. The learning algorithm's log probability is raised when a multiple hidden layer is added to the network, which is a strategy to improve true representational power. DBNs mainly rely on this massive increase in conjunction with the use of unlabelled data.

3.6.3. EfficientNet

The CNN models may be collectively referred to as the EfficientNet model, which is considered one of the state-of-the-art models having achieved an accuracy of 84.4% with 66 M variable in the ImageNet categorization task. The EfficientNet group's eight models, which range in number from B0 to B7, demonstrate that while accuracy increases dramatically, the number of calculated variables does not considerably increase with modeling number. While earlier CNN models employed the Rectifier Linear Unit (ReLU) activation function, EfficientNet integrates a new activation function called Swish. The purpose of deep learning architectures is to find better ways with fewer models. EfficientNet generates more efficient results than previous state-of-the-art models by scaling down the model and uniformly increasing depth, breadth, and resolution. Using the compound scaling technique, the initial step is to find a grid that illustrates the link between the several scaling dimensions of the baseline network under a set resource restriction.

The depth, breadth, and resolution parameters' proper scaling factor is determined in this way. Following that, these characteristics are used to scale the baseline network to the planned target network. Primarily, EfficientNet is based on the inverted bottleneck MBConv, which made its debut in MobileNetV2. However, because to its greater FLOPS budget, it is used a little more than in MobileNetV2. Bottlenecks are connected by direct connections, which connect a comparatively lower number of channels than expansion layers, because blocks in MBConv are composed of a layer that expands and subsequently compresses the channels. Deep separable convolutions in this design save almost a factor of k in computation when compared to standard layers; k is the kernel size that defines the width and height of the 2D convolution window. Apply the compound coefficient ψ together with the ideas given in Eq. (49) to provide consistent depth, breadth, and resolution scaling in compound scaling.

$$depth: d = \alpha^{\psi} \quad (49)$$

$$width: w = \beta^{\psi} \quad (50)$$

$$resolution: r = \gamma^{\psi} \quad (51)$$

$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1 \quad (52)$$

In this case, the constants $\alpha, \beta, and\ \gamma$ may be found using grid search. The user-defined coefficient ψ represents the maximum amount of resources that may be allocated to model scaling, while the network breadth, depth, and resolution are allotted these additional resources according to $\alpha, \beta, and\ \gamma$, respectively. FLOPS are proportional to $d, w^2, and\ r^2$ in a normal convolution process. Scaling the convolution network according to Eq. (49) to Eq. (52) raises the FLOPS of the network by around $(\alpha, \beta^2, \gamma^2)^{\psi}$ $(\alpha . \beta^2 \gamma^2)^{\psi}$ overall as convolution operations account for the majority of the computational cost in convolution networks.

This model is scaled in two stages using the compound scaling approach, starting from Baseline EfficientNet-B0:

Step 1: Grid search is carried out with ψ = 1 under the assumption that there are twice as many resources available, and the optimal values for $\alpha, \beta, and\ \gamma$ are discovered.

Step 2: The values of $\alpha, \beta, and\ \gamma$ are established as constants, and the basic network is expanded to produce EfficientNet-B1 through B7 by utilizing Eq. (49) to Eq. (52) with varying values of ψ.

3.6.4. SqueezeNet

A more compact CNN design that retains comparable accuracy with fewer parameters is called SqueezeNet. SqueezeNet is built using many techniques that are applied to the CNN base. First, filters measuring three by three are replaced by filters measuring one by one. Second, there are fewer input channels for the three-by-three filters. Lastly, to offer bigger activation maps for convolution layers, downsampling is deliberately included at the end of the network. Fire modules make up the majority of SqueezeNet's components. These modules are distinguished by squeezing convolution layers that use just 1 × 1 filters. The extend layer receives input from these layers and consists of a blend of 1 × 1 and 3 × 3 convolution filters. Together, these design decisions improve the network architecture and increase SqueezeNet's effectiveness and compactness.

Along with the deep learning techniques, the dense layer and the fully connected layer are connected and the results of the fully connected layer is given to the scaled Dot-Product attention layer.

3.7. Scaled Dot-Product Attention

For each given d, the function $a : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is known as a scalar attention function. A query is the first parameter and a key are the second argument of an. The real number $a(q, k)$ for $q, k \in \mathbb{R}^d$ shall be referred to as the scalar attention of q on k.

Given a finite ordered set K of keys $k_1, k_2, \dots, k_n \in \mathbb{R}^d$and an attention function $a : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, that may construct a function $a_K : \mathbb{R}^d \to \mathbb{R}^n$, which is known as the vector attention function associated with K and based on a, which in given in Eq. (53),

$$a_K(q) = \begin{bmatrix} a(q, k_1) \\ a(q, k_2) \\ \vdots \\ a(q, k_n) \end{bmatrix} \tag{53}$$

Next, call $a_K(q)$ which is q's vector attention on K. Assume that we have a function S_K: $\mathbb{R}^n \to \Delta^{n-1}$ for this set K of n keys, where Δ^{n-1} is the standard $(n-1)$-simplex in $\mathbb{R}^n$.

$$\Delta^{n-1} = \left\{ \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n \middle| \sum_{i=1}^{n} x_i = 1\ and\ \forall i\ x_i \geq 0 \right\} \tag{54}$$

Such a function is denoted as S_K, which stands for rescaling. Then define the rescaled vector attention function A_K: $\mathbb{R}^d \to \Delta^{n-1}$ based on a, K, and S_K by using a rescaling function.

$$A_K = S_K \circ a_K \tag{55}$$

Within the framework of "Attention Is All You Need," the writers characterize their scalar attention function as a "scaled dot product":

$$a(q, k) = \frac{q.k}{\sqrt{d}} \tag{56}$$

They employ what is known as a rescaling function, described as $Softmax_n$: $\mathbb{R}^d \to \Delta^{n-1}$, for each set K of keys $k_1, k_2, \dots, k_n$:

$$Softmax_n\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}\right) = \frac{1}{\sum_{i=1}^{n} \exp(x_i)} \begin{bmatrix} \exp(x_1) \\ \exp(x_2) \\ \vdots \\ \exp(x_n) \end{bmatrix} \tag{57}$$

This produces the following rescaled vector attention function $A_K : \mathbb{R}^d \to \mathbb{R}^n$:

$$A_K(q) = \frac{1}{\sum_{i=1}^{n} \exp\,(q.k_i/\sqrt{d})} \begin{bmatrix} \exp\,(q.k_1/\sqrt{d}) \\ \exp\,(q.k_2/\sqrt{d}) \\ \vdots \\ \exp\,(q.k_n/\sqrt{d}) \end{bmatrix} \quad (58)$$

A precise and dependable classification of tomato leaf diseases is the outcome of attention mechanisms, which sharpen the model's emphasis on important components and provide an efficient disease classification strategy to wrap up the technique.

4. Result and discussion

This section presents a comparison of the proposed model's results with those of existing techniques. For the implementation, the Python platform is utilized. Of the data, 70% are for training and 30% are for testing. In order to train the model on one subset and subsequently test its generalization ability, it must be divided into two subsets. The suggested approach for predicting tomato leaf disease is assessed using the plant village dataset [24].

4.1. Dataset Description

A substantial collection of photos of diseased plant leaves together with the labels that go with them can be found in the PlantVillage dataset. A variety of plant diseases, including leaf mould, bacterial spot, early and late blight, and more, are depicted in the collected photos. A considerable number of individuals have downloaded and accessed the dataset, demonstrating its popularity and usefulness. Images of sick plant leaves with labels accompanying them make up the content. Photos showing a variety of plants, including tomatoes, potatoes, and peppers, with various illnesses are included in the collection.

4.2. Overall comparison by varying the learning rate

The suggested model's performance measures are compared to those of other methods, such as CNN [16], GoogleNet [18], RCNN [21], and MobileNet [6**]**. Table 1 displays the comparison.

Table 1: Comparison of performance metrics for Learning rate 70%

Performance metrics	CNN [16]	GoogleNet [18]	RCNN [21]	MobileNet [6]	PROPOSED
Accuracy	0.994255	0.99542	0.958993	0.982057	0.99746
Precision	0.971274	0.977102	0.794963	0.910283	0.987302
Sensitivity	0.971274	0.977102	0.794963	0.910283	0.987302
Specificity	0.996808	0.997456	0.977218	0.990031	0.998589
F-Measure	0.971274	0.977102	0.794963	0.910283	0.987302
MCC	0.968082	0.974558	0.772181	0.900315	0.985891
NPV	0.996808	0.997456	0.977218	0.990031	0.998589
FPR	0.003192	0.002544	0.022782	0.009969	0.001411
FNR	0.028726	0.022898	0.205037	0.089717	0.012698

Table 1 presents a comparative analysis of several models' performance indicators, encompassing CNN, GoogleNet, RCNN, MobileNet, and the suggested model with a 70% learning rate. The suggested model performs better on a range of criteria. It outperforms CNN, GoogleNet, RCNN, and MobileNet with the greatest accuracy (0.99746), precision (0.987302), sensitivity (0.987302), specificity (0.998589), F-measure (0.987302), and Matthews Correlation Coefficient (MCC) of 0.985891. Furthermore, the suggested model has a minimal False Negative Rate (FNR) of 0.012698, a low False Positive Rate (FPR) of 0.001411, and an exceptional

Negative Predictive Value (NPV) of 0.998589. These findings imply that, in contrast to the other models assessed, the suggested model performs very well in reliably identifying and detecting characteristics.

Table 2: Comparison of performance metrics for Learning rate 80%

Performance metrics	CNN [16]	GoogleNet [18]	RCNN [21]	MobileNet [6]	PROPOSED
Accuracy	0.995254	0.996253	0.96632	0.989134	0.998168
Precision	0.97627	0.981266	0.831599	0.94567	0.990841
Sensitivity	0.97627	0.981266	0.831599	0.94567	0.990841
Specificity	0.997363	0.997918	0.981289	0.993963	0.998982
F-Measure	0.97627	0.981266	0.831599	0.94567	0.990841
MCC	0.973633	0.979184	0.812887	0.939634	0.989823
NPV	0.997363	0.997918	0.981289	0.993963	0.998982
FPR	0.002637	0.002082	0.018711	0.006037	0.001018
FNR	0.02373	0.018734	0.168401	0.05433	0.009159

A comparison of performance metrics for CNN, GoogleNet, RCNN, MobileNet, and the suggested model is shown in Table 2 for various models at an 80% learning rate. With remarkable performance across a range of criteria, the suggested model stands out. Compared to the other models, it obtains the greatest values of accuracy (0.998168), precision (0.990841), sensitivity (0.990841), specificity (0.998982), F-measure (0.990841), and Matthews Correlation Coefficient (MCC) of 0.989823. A low False Positive Rate (FPR) of 0.001018, a minimal False Negative Rate (FNR) of 0.009159, and a good Negative Predictive Value (NPV) of 0.998982 are also demonstrated by the suggested model. Compared to CNN, GoogleNet, RCNN, and MobileNet at the designated learning rate, these results highlight the proposed model's better capacity to reliably categorize and identify features.

❖ **Accuracy**

The suggested model consistently performs better than the other models at both learning rates, as can be shown by comparing the accuracy values from Tables 1 and 2. The suggested model attains an accuracy of 0.99746 in Table 1 at a learning rate of 70%, and enhances to 0.998168 in Table 2 with a learning rate of 80%. The suggested model is resilient and successful in effectively categorizing and processing pictures, as demonstrated by these numbers, which outperform the accuracies of CNN, GoogleNet, RCNN, and MobileNet in both tables. Comparing the accuracy metric is displayed in Figure 4.

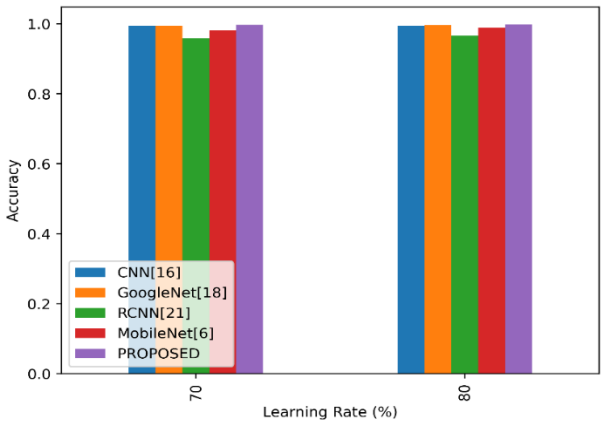


Figure 4: Comparison of the accuracy metric

❖ **Precision**

The suggested model consistently outperforms CNN, GoogleNet, RCNN, and MobileNet in terms of accuracy when comparing the precision values from Tables 1 and 2. This is true for both learning rates. The suggested model outperforms the other models with a precision of 0.987302, achieved at a learning rate of 70% in Table 1. At an 80% learning rate, Table 2's accuracy increases to 0.990841. These precision numbers show how well the model reduces false positives and improves the precision of positive predictions. Figure 5 shows the contrast of accuracy.

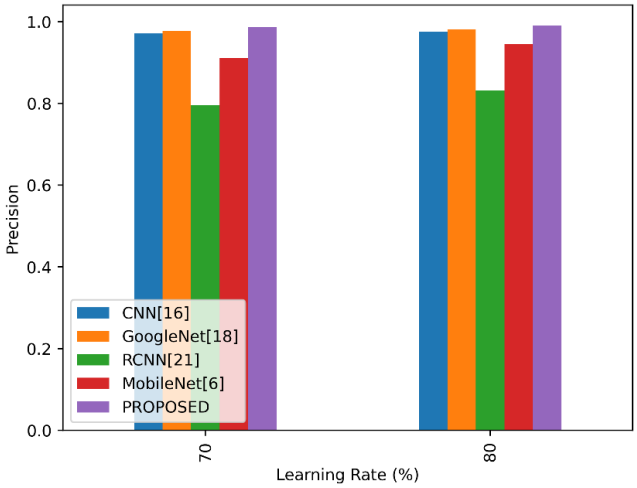


Figure 5: Comparison of the Precision metric

❖ **F-Measure**

When comparing the F-measure values of Tables 1 and 2, it is constantly seen that the suggested model performs better at both learning rates. The suggested model outperforms CNN, GoogleNet, RCNN, and MobileNet with an F-measure of 0.987302 at a learning rate of 70% in Table 1. Consequently, the F-measure in Table 2 improves even more to 0.990841 at a learning rate of 80%. Figure 6 shows the comparison of F-Measure.

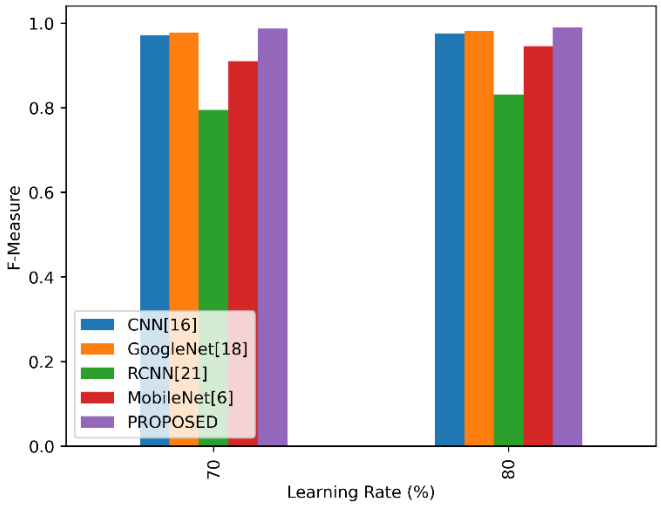


Figure 6: Comparison of the F-Measure metric

❖ **Sensitivity**

The suggested model continuously shows good performance across both learning rates when sensitivity values are compared across Tables 1 and 2. The sensitivity of the suggested model is 0.987302, above the values of CNN, GoogleNet, RCNN, and MobileNet, at a learning rate of 70% in Table 1. Moreover, Table 2 shows that the sensitivity rises to 0.990841 at a learning rate of 80%. Recall, or sensitivity, is a measure of how well the

model can distinguish true positive cases from all other true positive instances. Figure 7 presents the sensitivity comparison.

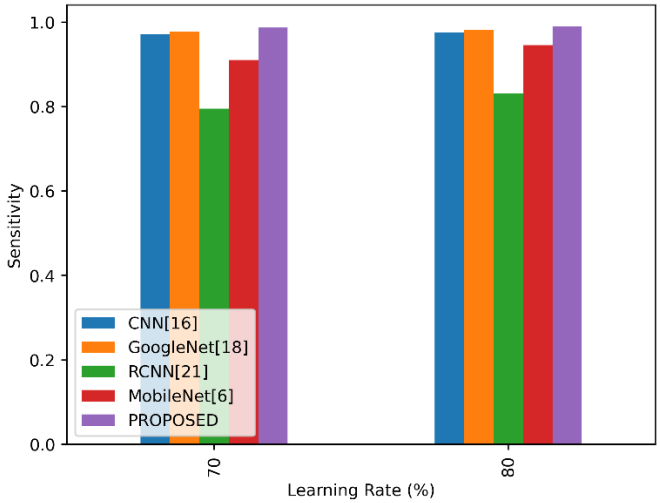


Figure 7: Comparison of the sensitivity metric

❖ **Specificity**

When specificity values are compared across Tables 1 and 2, the suggested model continuously exhibits superior performance for both learning rates. The suggested model's specificity, with a learning rate of 70% in Table 1, is 0.998589, surpassing CNN, GoogleNet, RCNN, and MobileNet. Similar to this, Table 2's specificity rises to 0.998982 with a learning rate of 80%. The model's specificity is a measure of how well it can distinguish between all real negative cases and negative instances. In Figure 8, the specificity is compared.

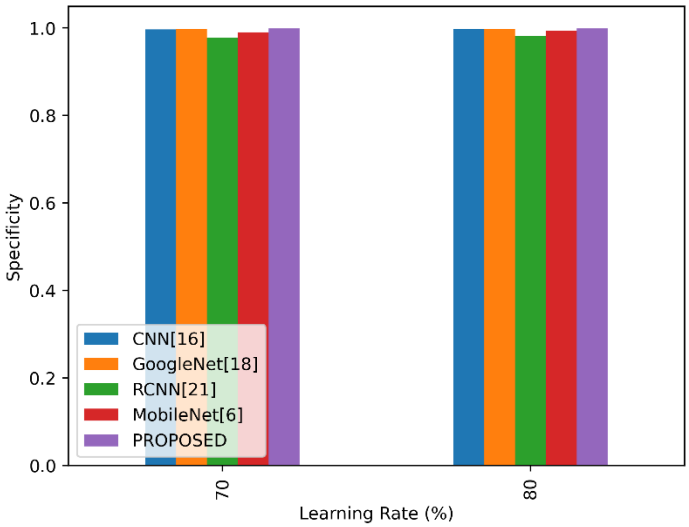


Figure 8: Comparison of the specificity metric

❖ **MCC**

When comparing the Matthews Correlation Coefficient (MCC) values across Tables 1 and 2, it is evident that the suggested model performs admirably at both learning rates. The suggested model has an MCC of 0.985891 at a learning rate of 70% in Table 1, which is higher than the values of CNN, GoogleNet, RCNN, and MobileNet. Similar to this, Table 2's MCC rises to 0.989823 at a learning rate of 80%. A thorough metric, the MCC considers values that are false positive, false negative, true positive, and true negative. Comparing the MCC is Figure 9.

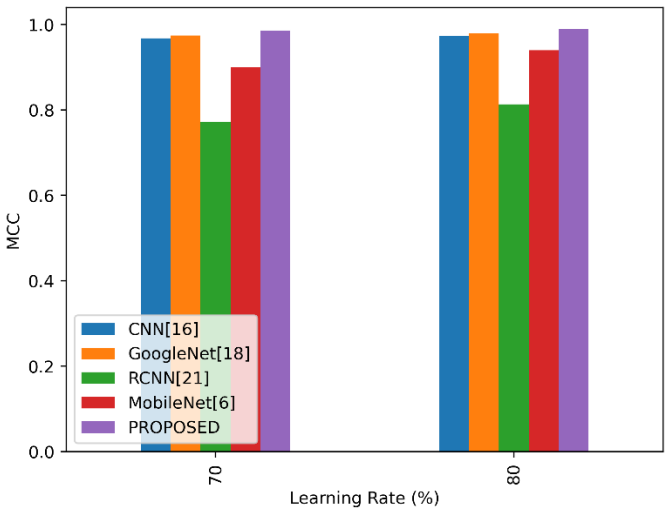


Figure 9: Comparison of the MCC metric

❖ **NPV**

The suggested model continuously performs well at both learning rates when comparing the Negative Predictive Value (NPV) values between Tables 1 and 2. The suggested model outperforms CNN, GoogleNet, RCNN, and MobileNet with an NPV of 0.998589 at a learning rate of 70% in Table 1. Likewise, in Table 2, for a learning rate of 80%, the NPV rises to 0.998982. NPV evaluates how well the model can distinguish genuine negatives from all cases that are projected to be negative.

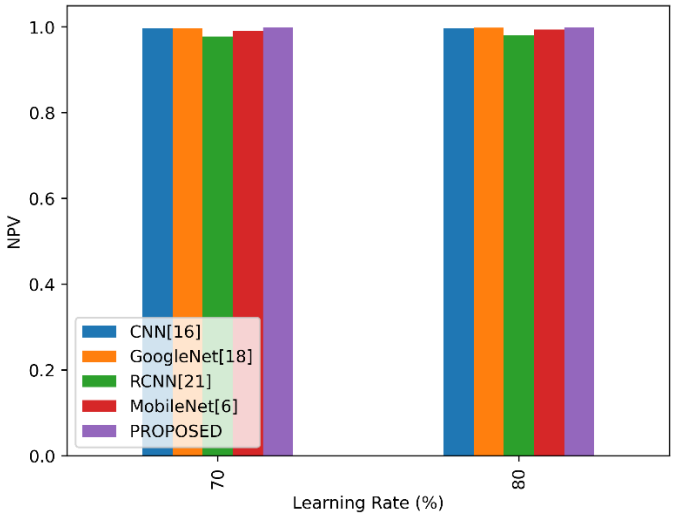


Figure 10: Comparison of the NPV metric

❖ **FNR**

The FNR for the suggested model, with a learning rate of 70% in Table 1, is 0.012698, a significant decrease from the values for CNN, GoogleNet, RCNN, and MobileNet. Similar to this, Table 2's FNR of 0.009159 is still low at an 80% learning rate. The fraction of true positive cases that the model mistakenly predicts as negative is measured by FNR. The suggested model's continuously low FNR values emphasize how well it works to minimize the exclusion of positive instances and how dependable it is at reliably collecting positive examples in image processing tasks at varying learning rates.

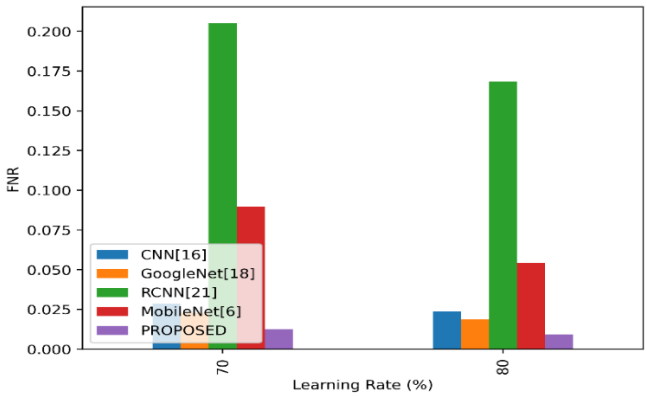


Figure 11: Comparison of the FNR metric

❖ **FPR**

At a 70% learning rate, the FPR of the proposed model is 0.001411, which is lower than the values for CNN, GoogleNet, RCNN, and MobileNet in Table 1. Likewise, for an 80% learning rate, Table 2's FPR of 0.001018 remains constant. False positive rate (FPR) measures the proportion of actual negative cases that the model incorrectly interprets as positive. in different learning rates, the proposed model's consistently low false positive rates (FPR) show how successful it is in lowering false alarms and ensuring accurate negative predictions in image processing tasks.

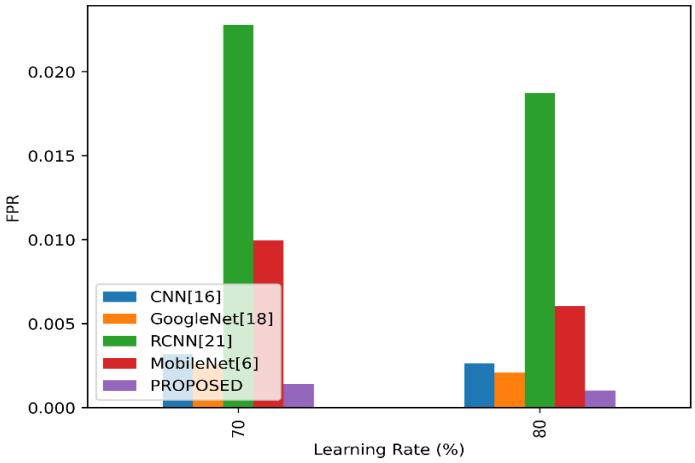


Figure 12: Comparison of the FPR metric

5. Conclusion

In conclusion, the suggested approach offers a thorough and sophisticated framework for the precise and trustworthy identification of illnesses affecting tomato leaves. By employing a methodical, multi-phase procedure that includes picture pre-processing, novel leaf localization, ROI selection, and feature extraction, the technique seeks to improve disease diagnosis accuracy. By integrating several neural network topologies with attention processes, NeuroFusionNet is used for illness categorization, which enhances the model's capacity to recognize complex patterns. Ultimately, by offering a reliable and effective approach for the diagnosis and categorization of tomato leaf diseases, the methodology makes a substantial contribution to the control of agricultural diseases. The utilization of state-of-the-art methods at every stage highlights the possible influence of this study for expanding the area of plant pathology and resulting in more efficient strategies for disease control in agricultural settings.

Acknowledgements

One of the authors (PSL) would like to thank Prof. Sanjay Bhargav for his constant encouragement and guidance , to the Head, Dean and the management of Anurag University, Hyderabad for their constant support and Dr. Ujwala for the academic discussions and suggestions.

References

[1] Trivedi, N.K., Gautam, V., Anand, A., Aljahdali, H.M., Villar, S.G., Anand, D., Goyal, N. and Kadry, S., 2021. Early detection and classification of tomato leaf disease using high-performance deep neural network. Sensors, 21(23), p.7987.

[2] Chen, H.C., Widodo, A.M., Wisnujati, A., Rahaman, M., Lin, J.C.W., Chen, L. and Weng, C.E., 2022. AlexNet convolutional neural network for disease detection and classification of tomato leaf. Electronics, 11(6), p.951.

[3] Kaur, P., Harnal, S., Gautam, V., Singh, M.P. and Singh, S.P., 2023. A novel transfer deep learning method for detection and classification of plant leaf disease. Journal of Ambient Intelligence and Humanized Computing, 14(9), pp.12407-12424.

[4] Abbas, A., Jain, S., Gour, M. and Vankudothu, S., 2021. Tomato plant disease detection using transfer learning with C-GAN synthetic images. Computers and Electronics in Agriculture, 187, p.106279.

[5] Saleem, M.H., Potgieter, J. and Arif, K.M., 2019. Plant disease detection and classification by deep learning. Plants, 8(11), p.468.

[6] Ashwinkumar, S., Rajagopal, S., Manimaran, V. and Jegajothi, B., 2022. Automated plant leaf disease detection and classification using optimal MobileNet based convolutional neural networks. Materials Today: Proceedings, 51, pp.480-487.

[7] Albattah, W., Nawaz, M., Javed, A., Masood, M. and Albahli, S., 2022. A novel deep learning method for detection and classification of plant diseases. Complex & Intelligent Systems, pp.1-18.

[8] Maeda-Gutiérrez, V., Galván-Tejada, C.E., Zanella-Calzada, L.A., Celaya-Padilla, J.M., Galván-Tejada, J.I., Gamboa-Rosales, H., Luna-Garcia, H., Magallanes-Quintanar, R., Guerrero Mendez, C.A. and Olvera-Olvera, C.A., 2020. Comparison of convolutional neural network architectures for classification of tomato plant diseases. Applied Sciences, 10(4), p.1245.

[9] Agarwal, M., Singh, A., Arjaria, S., Sinha, A. and Gupta, S., 2020. ToLeD: Tomato leaf disease detection using convolution neural network. Procedia Computer Science, 167, pp.293-301.

[10] Tiwari, V., Joshi, R.C. and Dutta, M.K., 2021. Dense convolutional neural networks based multiclass plant disease detection and classification using leaf images. Ecological Informatics, 63, p.101289.

[11] Thangaraj, R., Anandamurugan, S. and Kaliappan, V.K., 2021. Automated tomato leaf disease classification using transfer learning-based deep convolution neural network. Journal of Plant Diseases and Protection, 128(1), pp.73-86.

[12] Agarwal, M., Gupta, S.K. and Biswas, K.K., 2020. Development of Efficient CNN model for Tomato crop disease identification. Sustainable Computing: Informatics and Systems, 28, p.100407.

[13] Ahmad, I., Hamid, M., Yousaf, S., Shah, S.T. and Ahmad, M.O., 2020. Optimizing pretrained convolutional neural networks for tomato leaf disease detection. Complexity, 2020, pp.1-6.

[14] Basavaiah, J. and Arlene Anthony, A., 2020. Tomato leaf disease classification using multiple feature extraction techniques. Wireless Personal Communications, 115(1), pp.633-651.

[15] Ahmed, S., Hasan, M.B., Ahmed, T., Sony, M.R.K. and Kabir, M.H., 2022. Less is more: Lighter and faster deep neural architecture for tomato leaf disease classification. IEEE Access, 10, pp.68868-68884.

[16] Bhujel, A., Kim, N.E., Arulmozhi, E., Basak, J.K. and Kim, H.T., 2022. A lightweight Attention-based convolutional neural networks for tomato leaf disease classification. Agriculture, 12(2), p.228.

[17] Gadekallu, T.R., Rajput, D.S., Reddy, M.P.K., Lakshmanna, K., Bhattacharya, S., Singh, S., Jolfaei, A. and Alazab, M., 2021. A novel PCA–whale optimization-based deep neural network model for classification of tomato plant diseases using GPU. Journal of Real-Time Image Processing, 18, pp.1383-1396.

[18] Wu, Q., Chen, Y. and Meng, J., 2020. DCGAN-based data augmentation for tomato leaf disease identification. IEEE Access, 8, pp.98716-98728.

[19] Tan, L., Lu, J. and Jiang, H., 2021. Tomato leaf diseases classification based on leaf images: a comparison between classical machine learning and deep learning methods. AgriEngineering, 3(3), pp.542-558.

[20] Zhou, C., Zhou, S., Xing, J. and Song, J., 2021. Tomato leaf disease identification by restructured deep residual dense network. IEEE Access, 9, pp.28822-28831.

[21] Wang, Q., Qi, F., Sun, M., Qu, J. and Xue, J., 2019. Identification of tomato disease types and detection of infected areas based on deep convolutional neural networks and object detection techniques. Computational intelligence and neuroscience, 2019.

[22] Karthik, R., Hariharan, M., Anand, S., Mathikshara, P., Johnson, A. and Menaka, R., 2020. Attention embedded residual CNN for disease detection in tomato leaves. Applied Soft Computing, 86, p.105933.

[23] Chen, X., Zhou, G., Chen, A., Yi, J., Zhang, W. and Hu, Y., 2020. Identification of tomato leaf diseases based on combination of ABCK-BWTR and B-ARNet. Computers and Electronics in Agriculture, 178, p.105730.

[24] Dataset is taken from https://www.kaggle.com/datasets/emmarex/plantdisease dated on 01/12/2023.