Neural Computing and Applications (2022) 34:5513–5534
https://doi.org/10.1007/s00521-021-06714-z

ORIGINAL ARTICLE

An automatic plant leaf disease identification using DenseNet-121 architecture with a mutation-based henry gas solubility optimization algorithm

S. Nandhini[1] · K. Ashokkumar[1]

Received: 7 April 2021 / Accepted: 30 October 2021 / Published online: 3 January 2022

Abstract
Farmers are struggling to provide the fast-growing population with sufficient agricultural products, while plant diseases result in devastating food loss. The billions of dollars spent by agriculturists in disease management often result in poor disease control without any technical support. Advances in computer vision techniques help to detect plant pathogens at an earlier level with an adaptive algorithm designed through deep learning and machine learning techniques. In this paper, we present an efficient Mutation-based Henry Gas Solubility Optimization (MHGSO) algorithm to optimize the hyperparameters of the DenseNet-121 architecture. The hyperparameter optimization is mainly done to reduce the computational complexity and the error rate of the Convolutional Neural Network (CNN). This step helps the MHGSO optimized DenseNet-121 architecture to achieve a higher classification accuracy for classifying different plant images from the PlantVillage dataset. The experimental results achieved showed that the proposed model is capable of classifying 14 leaf classes present in the PlantVillage dataset with higher classification accuracy (98.7%) and stability. When tested with a field dataset with complicated backgrounds, the proposed MHGSO optimized DenseNet-121 architecture achieves accuracy, precision, and recall scores of 98.81%, 98.60%, and 98.75%, respectively.

Keywords Plant leaf disease identification · DenseNet-121 · Henry gas optimization algorithm · Convolutional neural network

1 Introduction

India is a country where the majority of the population depends upon agriculture [1] for its income. Agriculture is said to be the backbone of the Indian economy. India is the second-largest producer of fruits and vegetables in the world [2]. At the same time, India's economy faces a loss in crop yield due to the numerous issues faced by the farmers on a daily basis. Among them, one of the major issues the farmer faces is the loss of crop yield due to plant diseases. The diseases that affect the plants are often unseen while others reduce their productivity or destroy the crop entirely. The diseases of plants found with naked eyes are not always precise, and on the rural side locating plant disease experts is not easy.

If the diseases are identified at an early stage emergence of the pathogens can be controlled. The challenges [3] that often rise to automatic classification of leaf diseases are interclass similarities, complex image background, occlusion, color, pose, and brightness. The early diagnosis of plant disease is often impossible in different parts of the world due to inadequate resources. The novel image processing techniques [4–9] such as machine learning, computer vision, and deep learning techniques can be useful in identifying the plant disease at an early stage and observe their progression of the disease. The evolution of low-cost mobile phones makes it easy for the farmers to take an image of the diseased leaf and sent it to the leaf disease identification application for analysis. The plant leaf disease identification application analyzes the leaf disease images with the help of an intelligent algorithm (deep

✉ S. Nandhini
nandhiniseec@gmail.com

[1] Department of Computer Science and Engineering, Sathyabama Institute of Science and Technology, Chennai, India

learning and machine learning techniques) and provides the solution to control the prognosis of this disease.

Deep Learning has emerged in recent years and has shown major advancements in various fields of application such as medicine, agriculture, finance, health care, education, entertainment, etc. [10]. Deep learning provides many benefits and the ability to manage massive datasets is one of the main strengths. A CNN [7, 11] is a deep learning algorithm that is widely used for image processing tasks. It captures both the spatial and temporal features of the images by applying relevant filters. Transfer learning [12] is a subfield of machine learning which is mainly used to minimize the number of computational resources used, error rate, and the time taken for model development. Transfer learning is capable of transferring the knowledge of the model trained in large datasets to a model with a small dataset [13].

In CNN, transfer learning is mainly applied by freezing the first few convolutional layers of the network and training only the last layers to make a prediction [14, 15]. Although CNN architectures [16, 17] have achieved promising results in image processing tasks, the large number of hyperparameters [18–20] required to generate highly accurate results is of significant complexity. The values of the CNN hyperparameters have a crucial effect on the behavior and the performance of the CNN architectures. The hyperparameter value should be adjusted for each dataset since the hyperparameter that fits a particular dataset does not go well with the other dataset. The existing researches mainly selected the hyperparameter values using manual trial and error, random search, grid search, and tactful experts. Only a limited amount of studies [20–22] has analyzed the importance of these hyperparameters to obtain higher efficiency and considered it as an optimization problem. To optimize the computationally intensive CNN architecture, an automatic hyperparameter optimization architecture is proposed in this paper using a mutation-based Henry Gas Solubility Optimization (MHGSO) algorithm. The MHGSO algorithm helps to optimize the pre-trained DenseNet121 architecture which results in higher classification accuracy of the plant leaf disease classification. The main contributions of this paper are illustrated below:

- The proposed MHGSO optimized Densenet-121 architecture utilizes a transfer learning technique for a pre-trained architecture to classify different types of plant leave disease and the MHGSO algorithm optimizes the hyperparameters of the pre-trained DenseNet-121 architecture and offers improved performance.
- To improve the diversity of the HGSO algorithm and reach a globally optimal solution, a mutation subprogram is added. The mutation subprogram added enhances the randomization of the HGSO algorithm by increasing the search space.
- The proposed model is capable of identifying the 14 leaf disease classes with a classification accuracy of 98.7%. The extensive experiments conducted using the PlantVillage and the field dataset using different performance metrics show that the proposed approach is an efficient method for plant leaf disease classification.

The rest of this paper is arranged as follows. Section 2 presents the existing literature works of various authors in this field. Section 3 presents the different concepts used to form the proposed methodology along with the MHGSO algorithm formulation. Section 4 presents the dataset description along with the working of the proposed model. Section 5 provides the experimentation conducted on the proposed methodology to evaluate its effectiveness using different performance metrics. Section 6 concludes the research.

2 Review of related works

A. Athiraja et al. [4] provided a solution to the banana disease diagnosis using both machine learning and computer vision techniques. The noise removal process is done by standardization techniques and a soft coring filter. After the feature extraction process is completed, two classification techniques are used namely Case-Based Reasoning (CBR) and Adaptive Neuro-Fuzzy Inference System (ANFIS). However, the ANFIS approach suffers from the curse of dimensionality and high computational cost. Rakasekaran Thangaraj et al. [5] developed a transfer learning-based DCNN to identify the different classes of tomato leaf diseases. The model is capable of classifying both the images obtained from the dataset and the real-time images. They used the transfer learning concept to reduce the time taking for model development and to lower the computational resource consumption. Using transfer learning they developed a modified Xception model which is an extension of the Inception model using 36 separable convolutional layers for feature extraction.

David Argüeso et al. [6] developed a Few Shot Learning (FSL) architecture to learn the novel leaf disease classes from small datasets. The FSL model uses three modules namely model initialization, metric learning, and data hallucination for image classification. The FSL architecture built in this paper utilizes a siamese network with a triplet loss along with a multiclass Support Vector Machine classifier. The convergence problems that normally occur in the FSL are overcome here using the triple loss-based Siamese networks. Miaomiao Ji et al. [7] designed a

UnitedModel based multiple CNN to identify the common diseases that affect the grape plant such as esca, black rot, and *isariopsis* leaf spot. The complementary discriminatory features can be efficiently extracted via multiple CNN. When evaluated on the hold-out plant Village dataset, the model provides an average validation and test accuracy of 99.17% and 98.57%.

Abdul Waheed et al. [8] proposed an optimal Dense-CNN architecture utilizing the Densenet architecture for corn leaf disease identification and classification. The optimized DenseNet model is mainly designed to overcome challenges such as the increasing number of CNN parameters and the network size. The Dense-CNN architecture provides an accuracy of 98.06% and it is said to be computationally cost-effective since it reduces the parameter size up to 77,612. Chongke Bi et al. [23] introduced the high-precision, stable, and cost-effective MobileNet classification architecture for apple leaf diseases. The model is claimed to be inexpensive and stable since it is used on mobile devices and it can be analyzed by anyone without any experience in the agriculture field. The overall accuracy and average time taken to classify each instance using the MobileNet model is 73.50% and 0.22 s.

Ramar Ahila Priyadharshini et al. [24] presented a modified LeNet based CNN architecture to identify the different maize leaf diseases. The LeNet architecture is used to automatically extract the features from the input image in a systematic manner. They used a kernel size of 3×3 to attain a classification accuracy of 97.89%. Mohit Agarwal et al. [25] presented an efficient CNN model utilizing eight hidden layers for tomato leaf disease classification from the PlantVillage dataset. To improve the performance of the classifier, the image's brightness is altered using a random value after the image augmentation process in the preprocessing stage. This model yields an accuracy of 98.4% with less storage space.

Several Researchers have used metaheuristic algorithms for plant leaf disease diagnosis and some of them are delineated below. Diana Andrushia et al. [11] presented an Artificial Bee Colony (ABC) optimization algorithm for feature selection. To eliminate the noise and background, the input images are preprocessed. After the color, texture, and shape features are extracted, the ABC algorithm is used for optimal feature set extraction. Afterward, these results are passed to the Support Vector Machine (SVM) classifier for identifying the foliar diseases in grapes. Cristin et al. [26] presented a Rider Cuckoo Search (RCS) algorithm for training the Deep Belief Network (DBN). The input leaf disease image is preprocessed and segmented using a piecewise Fuzzy C-Means (piFCM). After segmentation, the texture features are extracted using information gain, entropy, and Histogram of Gradient (HOG). Existing approaches [27, 28], on the other hand, perform better when applied to different disease classes of the same crop as opposed to different crops, and they are not used on datasets with complex backgrounds. To solve this problem, the proposed study aims to find disease variations in more than 7 crops that can be found in both the public dataset (PlantVillage) and the field dataset with complex backgrounds. The summary of these works along with the crop species and disease identified is present in Table 1.

3 Methods used

3.1 Convolutional neural network

The CNN structure and functionality resemble the visual cortex [29] and it is widely used for image processing and analysis. To classify the raw input data, the CNN network selects the most appropriate filters for classification. Since the feature extraction and identification stages are integrated into CNN, the computational cost associated with other machine learning techniques [30] to perform this procedure is eliminated. The three main components of the CNN are the convolutional layer, pooling layer, and fully connected layer. The convolutional layer is the main component of the CNN and it comprises of the learnable kernels which are interlinked with the local regions of the input dimension.

The receptive fields are the regions present in the input dimension which is influenced by a specific CNN filter. Using the same kernel, the local features of the input are extracted using the filter. Multiple features are extracted by sliding the filter along the different widths and heights of the input dimension. Each filter has a different weight vector associated with it. At last, using an activation function, the output feature maps are generated. The Rectified Linear Unit (ReLU) is the activation function used here because of its less computational requirement and fast training offered. The major advantage offered by the ReLU activation function is that it does not activate every neuron at the same time which overcomes the dead neuron problem. The ReLU activation function is defined as $f(a) = \max(0,a)$. ReLU directly outputs the input or else it is zero. In this way, the elemental non-linearity is applied to the convolutional input. The output of the convolutional layer is obtained using the below equation:

$$a_x^{k+1}(y) = \mathrm{Re}LU\left(\sum_{y=0}^{N} \omega_x^k * A^k(y) + \beta_x^k\right) \quad (1)$$

where x represents the filter, k represents the layer, β indicates the bias, ω indicates the weight, N is the kernel width, $A^k(y)$ represents the y^{th} local region in the k^{th} layer,

Table 1 Summary of the literary works

Paper	Methodology applied	Crop species and diseases identified	Limitation	Accuracy
A.Athiraja et al. [4]	ANFIS and CBR	Anthracnose, Panama Wilt, Leaf spot, Crown root virus disease, and Cigar end tip root of the banana plant	Curse of dimensionality and high computational cost	97.5%
Rakasekaran Thangaraj et al. [5]	Xception model	Early Blight, Late Blight, Target spot, bacterial spot, leaf mold, mosaic virus, septoria leaf spot, spider mite, two-spotted spider mite, yellow leaf curl virus, and target spot disease in tomato plant	Classifies the diseases present in a single crop alone	99.55%
David Argüeso et al. [6]	Siamese FSL with triplet loss	Scab, black rot, and cedar rust diseases in apple	The Siamese network consumes more training time when compared to other networks	90.0%
Miaomiao Ji et al. [7]	UnitedModel based multiple CNN	Esca, black rot, and isariopsis leaf spot diseases in grape plant	The early stopping strategy doesn't utilize the available training data well. This model is only applied to differentiate three diseases of the same crop	Validation accuracy-99.17% Test Accuracy-98.57%
Abdul Waheed et al. [8]	Dense-CNN	Common rust, northern leaf blight, and Cercospora leaf spot in corn	Focuses on three diseases present in a single crop alone	98.06%
Chongke Bi et al. [23]	MobileNet	Alternaria leaf spot and rust disease in the apple crop	The dataset used comprises a low number of images	73.50%
Ramar Ahila Priyadharshini et al. [24]	LeNet based CNN architecture	Common rust, gray leaf spot, and northern leaf blight of maize leaf	The LeNet architecture is prone to overfitting	97.89%
Mohit Agarwal et al. [25]	Lightweight CNN	Bacterial Spot, Late Blight, Septoria Leaf Spot, Tomato Mosaic, and Yellow curled of the tomato plant	Accuracy decreases for images with complex backgrounds	98.4%
Diana Andrushia et al. [11]	ABC optimization algorithm and SVM	Leaf blight, Esca, and black rot diseases of grape	Premature convergence of the ABC algorithm	93.01%
Cristin et al. [26]	DBN, piFCM, and RCS	–	Higher time complexity associated with segmentation and feature extraction techniques	0.877

and $a_x^{k+1}(y)$ represents the j^{th} neuron input in the x frame present in layer $k+1$ respectively. The convolutional operation is represented using * in the above equation and it computes the dot product of both the local regions and the kernel. Figure 1a shows a pictorial representation of the convolution operation for further clarification, and the figure is self-explanatory.

After the convolutional layer, the pooling layer is present which performs the downsampling operation. The downsampling operation decreases the spatial dimension of the features without altering their depth and it is represented using Fig. 1b. In this way, the correlation features are integrated into one by simultaneously reducing the number of network parameters and computational load. The output of the max-pooling layer is a feature map that comprises the most crucial features of the previous feature map. The max-pooling operation is explained using the below equation:

$$m_x^{k+1}(y) = \max_{\substack{(y-1)\alpha+1 \le n_i \le y\alpha \\ (y-1)\gamma+1 \le n_j \le y\gamma}} \left(a_x^k(n_i, n_j)\right) \tag{2}$$

Here, $a_x^k(n_i, n_j)$ is the value of the neuron (n_i, n_j), α and γ are the coefficients that represent the height and width of the pooling layer, and $m_x^{k+1}(y)$ is the y^{th} neuron value in the $(k+1)^{\text{th}}$ pooling layer function. Figure 1c shows how to apply the max-pooling operation to an image using a 2×2 filter and a stride size of 2. The maximum value in each sub-region is returned by the max-pooling filter [31]. The classification part uses a fully connected layer that functions similarly to a Multi-Layer Perceptron (MLP) network that receives the entire feature maps to compute the

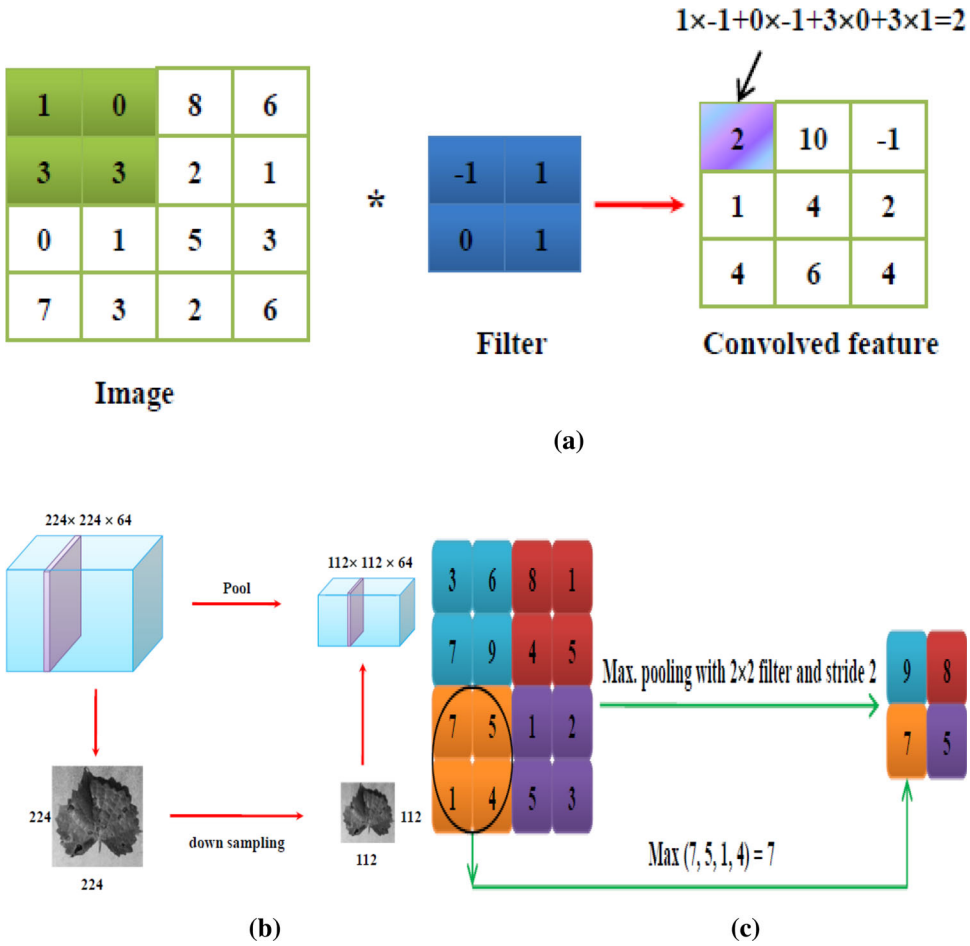


Fig. 1 Pictorial illustration of convolution and max pooling operations. (**a**) Convolution operation. (**b**) downsampling operation. (**c**) max-pooling operation

probability value of each class. As the name implies every neuron is connected to its previous layer. The softmax function is the activation function present in the output layer which computes the probability of each output leaf disease class for the given input image in a set of N possible output classes.

$$\sigma(o)_y = \frac{e^{o_y}}{\sum_{m=1}^{M} e^{o_m}} \tag{3}$$

Here N is the total number of classes present in the multiclass classification problem, e^{o_y} is the standard exponential function that operates on the unscaled output of the y^{th} neuron, and $\sigma(o)_y$ represents the probability value of the y^{th} actual output class. The conventional CNN architecture for leaf disease classification is presented in Fig. 2.

3.2 Henry gas solubility optimization algorithm

The HGSO algorithm [32] mainly mimics the behavior of Henry's law [33]. The Henry gas law was proposed by William henry in the year 1803. Henry's law constant states that in a constant temperature, the amount of gas dissolved in the liquid is directly proportional to partial pressure above the liquid. The henrys law is formulated using the below equation:

$$G_{sol} = H \times G_{press} \tag{4}$$

In this way, the solubility of low soluble gases in liquids can be established. The solubility of the liquid is mostly affected by two factors namely temperature and pressure. The solubility of the solid is high whereas the solubility of the gas is very low. When the pressure increases, the solubility of the gas also increases. The HGSO algorithm is mathematically formulated using the following steps:

i. *Initialization*

The population size M represents the number of gases and the position of these gases is initialized using the below equation:

$$A_x(k+1) = A_{\min} + rand \times (A_{\max} - A_{\min}) \tag{5}$$

Here A_x denotes the x^{th} gas in population M, *rand* is a random number whose value falls between 0 and 1, k is the time taken for each

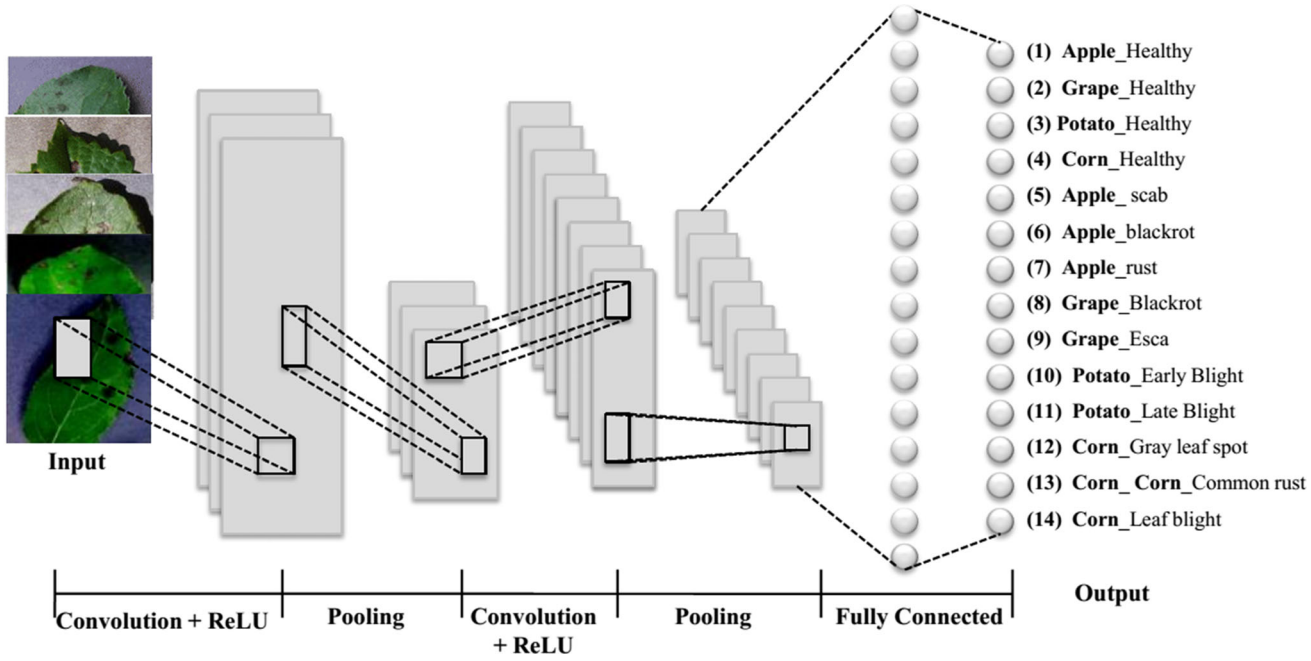


Fig. 2 CNN architecture

iteration, and the boundaries of the problem are represented using A_{max} and A_{min}. The gas value x, partial pressure $D_{x,y}$ of gas x in cluster y, henrys constant value of type y ($H_y(k)$), and the initial constant value for type y (K_y) is initialized using the below equation:

$$H_y(k) = s_1 \times rand(0,1),\ D_{x,y} = s_2 \times rand(0,1),\ S_y = s_3 \times rand(0,1) \tag{6}$$

where s_1, s_2, s_3 are the constants whose values equal 5E-20, 1E + 02, and 1E-02.

ii. *Clustering*

The agents in the population are partitioned into an equal number of clusters based on the gas types. The gases present in the same cluster have equivalent properties and the same henry constant value(H_y).

iii. *Evaluation*

Every cluster y is estimated to find the best gas that obtains the maximum equilibrium state when compared to the remaining gas of its type. To obtain the optimal gas from the whole swarm, the gas values are ranked.

iv. *Updating the Henrys coefficient*

Based on the below equation the Henry's coefficient is updated.

$$H_y(k+1) = H_y(k) \times \exp\left(-S_y \times \left(1/R(k) - {}^{1}/_{R^{\theta}}\right)\right);\ R(k) = \exp\left(-{}^{k}/_{i}\right) \tag{7}$$

where R is the temperature, i is the number of iterations and R^{θ} is the constant value used. The temperature constant value assigned is taken as 298.15.

v. *Solubility update*

Depending upon the below equation, the solubility of the gas particle is updated.

$$R_{x,y}(k) = C \times H_y(k+1) \times D_{x,y}(k) \tag{8}$$

Here, $R_{x,y}(k)$ represents the solubility of gas x in cluster y, C is the constant value, and $D_{x,y}(k)$ is the pressure of the gas x in cluster y.

vi. *Position update*

The position of the gas particle is updated using the following equation

$$A_{x,y}(k+1) = A_{x,y}(k) + G \times rand \times \mu \times (A_{x,best}(k) - A_{x,y}(k)) + G \times rand \times \beta \times (R_{x,y}(k) \times A_{best}(k) - A_{x,y}(k)) \tag{9}$$

$$\mu = \eta \times \exp\left(-\frac{G_{best}(k) + \alpha}{G_{x,y}(k) + \alpha}\right),\ \alpha = 0.05 \tag{10}$$

Here $A_{x,y}(k+1)$ represents the position of the gas x in cluster y, $A_{x,best}$ is the best gas x present in

cluster y, μ is the capacity of the gas y in cluster x that helps to interact will interact with other gases in the cluster, A_{best} is the best gas in the whole population, β is the influence of remaining gases on gas x, $G_{x,y}$ is the fitness value of the gas x in cluster y, and G_{best} is the fitness of the best gas value in the entire population. The β value is a constant and its value equals one. The parameter that controls both the tradeoff between the exploration and exploitation phases in the entire population is $A_{x,best}$ and A_{best}.

vii. *Avoidance of local optima*

The step shown below helps the algorithm from avoiding the local optima. The worst agents(P_V) are chosen and ranked based on the below equation.

$$P_V = P \times (rand(j_2 - j_1) + j_1), \; j_1 = 0.1 \, and \, j_2 = 0.2 \tag{11}$$

where P represents the number of search agents used.

viii. *Worst agent position update*

$$H_{x,y} = H_{\min(x,y)} + rand \times (H_{\max(x,y)} - H_{\min(x,y)}) \tag{12}$$

Here $H_{x,y}$ represents the position of gas x present in cluster y, H_{max}, and H_{min} are the boundaries of the hyper-parameter optimization problem. The flowchart of the HGSO algorithm is presented in Fig. 3.

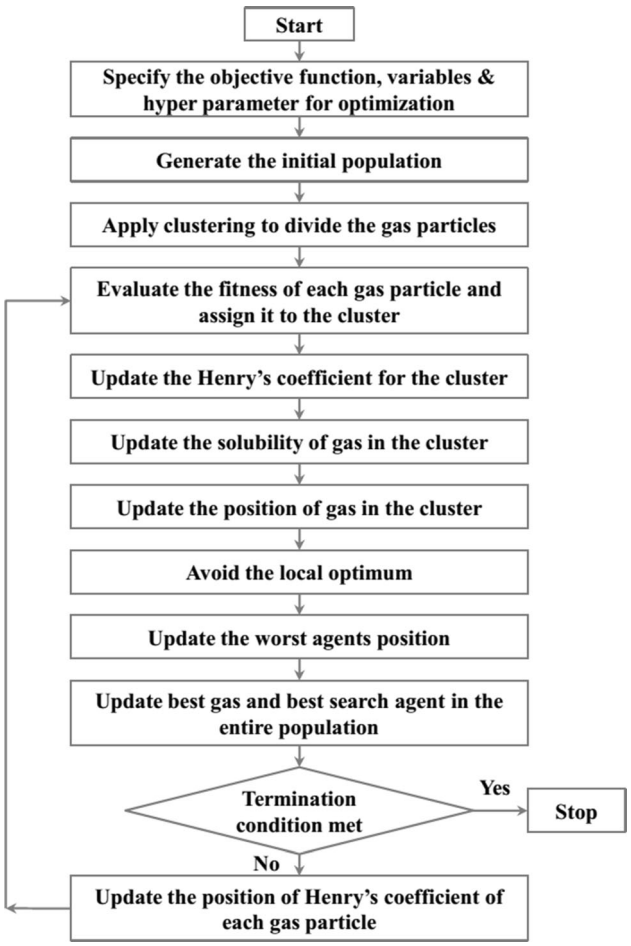


Fig. 3 Flowchart for the henry gas optimization algorithm

3.3 Formulation of M-HGSO Algorithm

If the diversity of the HGSO algorithm is low means, the HGSO algorithm struggles to reach the global optimum. This is a common problem encountered by most of the discrete optimizing problems. To overcome this problem in the HGSO algorithm, a mutation-based HGSO algorithm (M -HGSO) is proposed in this work. This algorithm differs from traditional HGSO in such a way that a mutation sub-program is added to strengthen the randomization of the algorithm, which simultaneously increases the search direction. A threshold condition is introduced in the mutation subprogram, as shown in the below equation:

$$TV = (1 - (x - 1)/MG - 1))^{\frac{1}{\psi}} \tag{13}$$

Here, TV represents the threshold value, MG represents the maximum number of generations, and ψ is the mutation factor. The mutation subprogram is called whenever a random number is below the TV value. This creates a new position randomly between the two boundaries of the problem (lower bound($Blower$) and upper bound($Bupper$)). Using Eqs. (14–17), the boundary values are computed. The new position is selected when its fitness value is better than its old position. The flowchart of the MHGSO algorithm is presented in Fig. 4.

$$\delta a = TV \times (A_{\max} - A_{\min}) \tag{14}$$

$$B_{lower} = A_x - \delta a \tag{15}$$

$$B_{upper} = A_x + \delta a \tag{16}$$

$$\left[B_{lower} \, B_{upper}\right] \in \left[A_{\min} A_{\max}\right] \tag{17}$$

3.4 Transfer learning

In the present era, most researchers are keen to transfer information from one domain to another. This is accomplished through the use of transfer learning [12], which saves time and effort while maintaining the accuracy of tested models. This saves money on the expensive GPU necessary to train the model. As the name implies, transfer learning occurs when a machine learning model is used to

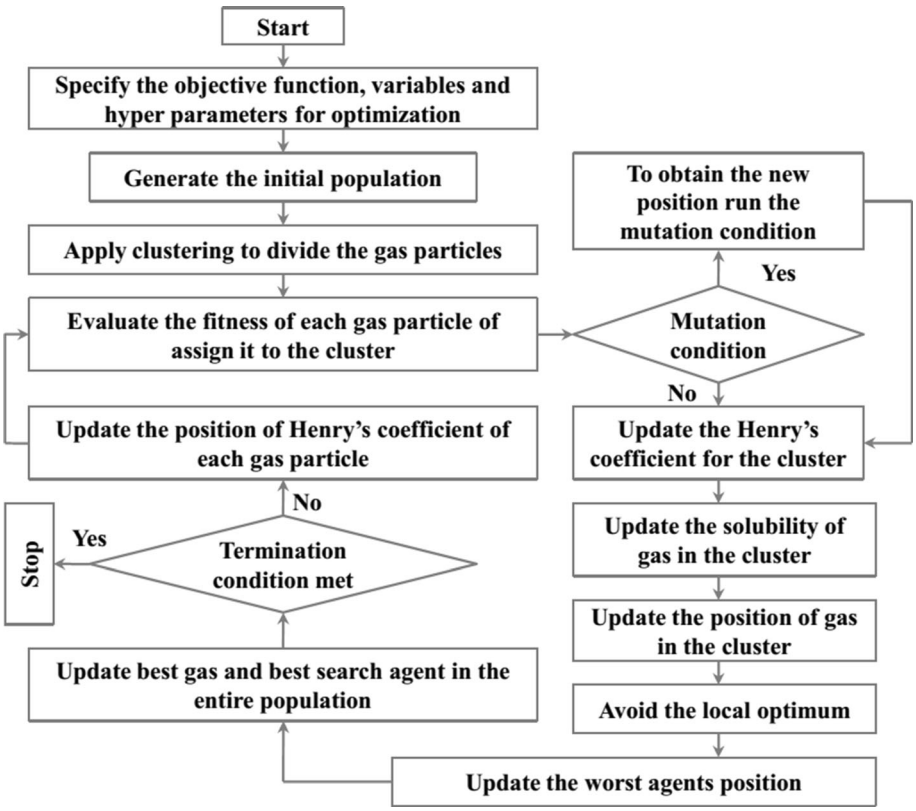


Fig. 4 Flowchart for the M-HGSO algorithm

solve one problem and then utilized as a starting point to solve another. It is typical practice to employ a deep learning model trained on a large data set, such as ImageNet, as a transfer learning model for a new model. Because training machine learning models requires time and effort, it is primarily used to speed up the training process and increase performance.

In deep learning, the image classification task is solved using transfer learning [13] to reduce time consumption and make the learning process faster for the current task. To provide solutions to the computer vision tasks and avoid developing a model from scratch, transfer learning uses the patterns that have been used by the pre-trained architectures for solving different computational problems. Using a pre-trained network is one of the most common methods for handling small datasets. A pre-trained network is initially trained using a massive dataset for different tasks such as image processing, text recognization, audio signal processing, speech recognition, etc. After the training process is completed its architecture and weight are maintained. The features trained using the pre-trained network serve as a normal visual model when the input dataset is large enough. Even if the pre-trained network is trained with completely different image features the model can be still useful for novel image classification tasks [10]. In this way, an accurate CNN model can be built and it can be both time-saving.

Most of the transfer learning-based pre-trained architectures solely depend upon the CNN. The conventional CNN comprises of two parts the convolutional base and the classifier part. The feature extraction is done using the convolutional base and the classifier part classifies the output based on the features extracted. Transfer learning for a pre-trained network is accomplished using two ways: feature extraction and fine-tuning. When the pre-trained DenseNet-121 architecture is reused in this work, the original classifier in it is replaced with a novel classifier that suits our need for plant leaf disease classification. In the feature extraction phase of the pre-trained network, the convolutional base is used to extract the novel dataset features and train a new classifier using the feature extraction outputs. The fine-tuning stage is the exact opposite of the feature extraction phase.

In the fine-tuning stage, the last layers of the convolutional base are frozen and this scenario can be taken as a complex case which deals with the train or freeze trade-off. Here, the convolutional base is used as it is and the outputs are fed to the classifier. A pre-trained architecture is used here for a stable feature extraction task and it can be quite useful when we have limited computational power or training using a small dataset. To prevent the weight from being updated during the training process, the freezing mechanism is used. After the feature extraction process is completed, the unfrozen layers are retrained by merging

them with the novel classifier developed. The transfer learning process is summarized as follows. Initially, a pre-trained architecture is taken and its classifier base is removed. In the next step, the pre-trained models' convolutional base is frozen and a novel classifier is added at the top of it. Depending upon the pre-trained network, unfreeze some of the layers present in the convolutional base. At last, both the novel classifier along the unfrozen layers are equally trained. Figure 5 visually depicts the pre-training model's fine-tuning procedure in terms of dataset size and similarity.

3.5 Proposed M-HGSO optimized Densenet-121 architecture for leaf disease classification

Applying the transfer learning concept to a pre-trained CNN architecture (DenseNet 121), the proposed M-HGSO based DenseNet-121 architecture has been formulated. To enhance the performance of the DenseNet-121 architecture, a hyperparameter optimization using the M-HGSO algorithm is proposed. After the algorithm outputs the optimal value for these hyperparameters, the transfer learning technique is applied to train the DenseNet-121 architecture. The architecture is evaluated using a separate test set after the training process is completed. Both the training and validation dataset is used to obtain the optimal values for the DenseNet-121 hyperparameters after training. The test set is evaluated using the fully trained DenseNet-121 architecture. The Proposed method comprises three phases namely data processing, hyperparameter analysis, and training as shown in Fig. 6. The description of these stages is presented in the next section in detail. The proposed plant leaf disease model is deployed in the cloud which can be automatically controlled via a cloud application. The farmers can take a snap of the diseased leaf using their mobile phones and update it on the cloud server. The data received at the cloud is processed using the proposed model by comparing it with the plant leaf images from the PlantVillage dataset stored in the cloud. After comparison, the leaf disease is accurately identified and given to the farmers via the cloud.

4 Methods and materials

4.1 Dataset description

For identifying the different types of plant leaf diseases we are using the PlantVillage dataset [34, 35]. The dataset comprises thirty-nine different classes of plant leaf images without any complex background. Hughes et al. [34] are the co-founders of the PlantVillage online platform which is mainly dedicated to crop welfare. The images in the PlantVillage dataset were obtained from the Land Grant university in the USA. The leaf was rotated at an angle of 360 degrees before taking the image and the images were captured under full sunlight. Since the leaves of the corn are very wide, they could not fit under a single frame while retaining a high-resolution constraint. In this scenario, different sections of the same leaf are captured. After the images are received, they are edited by cropping away the unnecessary background details and changing the

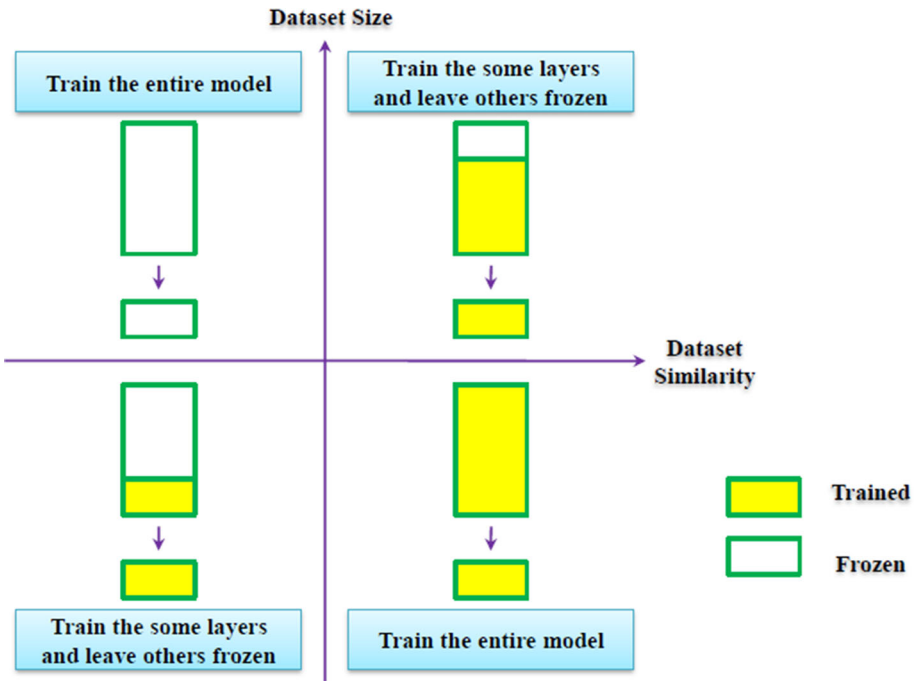


Fig. 5 Fine-tuning the pre-trained model in terms of structure and training

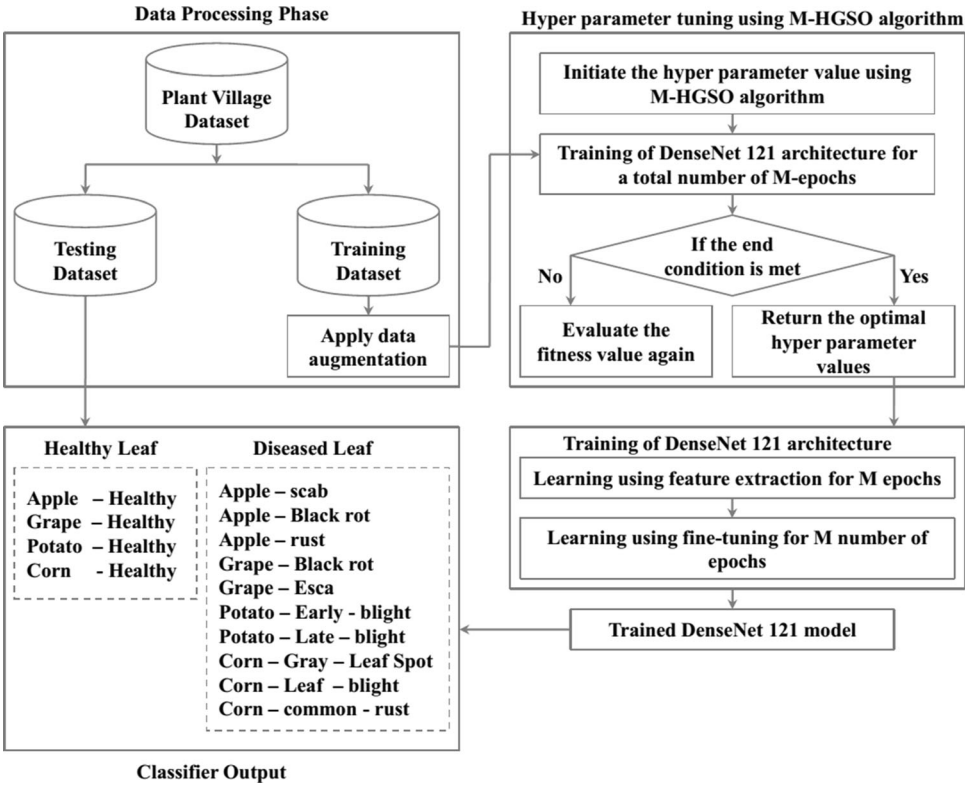


Fig. 6 Proposed M- HGSO optimized Densenet-121 architecture

orientation of the leaf such as the tip side pointed upwards. Table 2 describes the different leaf classes present in the PlantVillage dataset.

The proposed methodology is also evaluated with a field dataset comprising of complex backgrounds. The images in the dataset are obtained both from the agricultural farms in Tamilnadu and the Digipathos dataset [36]. This dataset is mainly taken to simulate the proposed work in real-time scenarios where the user uploads the image taken by them under natural conditions. The diseased leaf images were taken in the early stage when there is a presence of discolorations, round circles, or blobs. The images contain both complex (with many leaves or plants in the background) and normal backgrounds. The Sony Cybershot DSC-HX400V 20.4MP Digital Camera was used to capture these images. Light sources were also used to remove the obstacles and also the shadow of other leaves that distort the diseased images. The dataset consists of 13,676 images that belong to different leaf classes namely Papaya Mosaic Virus, Papaya Anthracnose, Papaya Healthy, Melon powdery mildew, Melon Mosaic, Melon healthy, Citrus Scab, Citrus canker, Citrus Greasy Spot, Citrus healthy, Grapevine bacterial canker, Grapevine powdery mildew, and Grapevine healthy. The dataset description of the field dataset is provided in Table 3.

4.2 Data processing

The details of the number of images present in each leaf class in the PlantVillage dataset are illustrated using Table 3. Table 3 also provides the details of the number of images after augmentation along with the images present in the training and testing dataset. Some classes in the dataset such as grape-healthy, potato-healthy, apple_scab, apple_blackrot, Corn_grayleafspot, and corn leaf blight consist of 152, 630, 621, 275, 513, and 985 images. The classes with a fewer number of images create a data imbalance issue. Not every machine learning classifier is capable of handling these types of data imbalance issues well. This type of information available in the dataset makes the classifier categorize the prominent classes and leaving out the minor ones.

Hence to overcome the data imbalance issue, the images in the grape-healthy, potato-healthy, apple_scab, apple_blackrot, Corn_grayleafspot, and corn leaf blight classes have been increased by randomly copying some already existing images and pasting them after augmentation. After this step, the dataset becomes balanced and each class has a minimum number of 1000 images. The balanced PlantVillage dataset is transformed into three sets: testing, training, and validation datasets. The dataset is divided into three parts (training, validation, and testing) in a 7:2:1 ratio. The number of images present in the training, validation, and the testing dataset is 11,205, 3112, and

Table 2 Description of different leaf disease classes present in the PlantVillage dataset

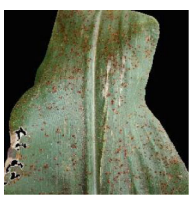
Name of the leaf disease	Image	Cause	Symptom	Impact
Apple scab		*Ascomycete fungus*	Light green irregular shaped lesions	Lesions on fruit, heavily infected fruit, and foilage fall
Apple black rot		*Botryosphaeria obtusa*	Purple margin and yellow to brown leaf center	Blossom end rot, fruit infection, and mummification of fruits
Cedar apple rust		*Gymnosporangium juniperi-virginianae Schwein* fungi	Circular, yellow spots on leafs upper surface, and a brownish cluster of threads around the yellow leaf spot	Reduces yield, fruit blemish, and leads to weakening
Grape Black rot		ascomycetous fungus, *Guignardia bidwellii*	Black fringes, small brown circular lesions, and brown patches in the leaf	Infection and mummification of grapes
Grape Esca		Fungi such as *Phaeoacremonium aleophilum, Phaeomoniella chlamydospora,* and *Fomitiporia Mediterranea*	Dark red or yellow stripes in leaves	Dryout and dieback of entire grapevine
Potato Early Blight		*Alternaria solani* fungi	Dark brown and faint concentric lesions which are oval or angular in shape	Results in higher defoliation and increases the chance of tuber infection
Potato Late Blight		*Phytophthora infestans* fungi	The broad yellow halo surrounding the affected lesions and water-soaked spots in tips and edges of lower leaves,	Significant crop loss
Corn Gray leaf spot		*Cercospora zeae-maydis* fungi	Gray lesions with chlorotic halos, block-shaped spots, and tan to brown coloration	Abrupts the photosynthesis process, stalk roots and causes higher damage during the pollination period
Corn Common rust		*Puccinia sorghi* fungi	Oval and elongated shaped dark reddish-brown pustules	Affects young leaves, chlorosis, and leaf death

Table 2 (continued)

Name of the leaf disease	Image	Cause	Symptom	Impact
Corn Leaf blight		*Setosphaeria turcica* fungi	Gray-green lesions that eventually turn pale gray or tan. The lesion size varies from small to large	Results in blighted leaves and causes death of plants

1245 respectively. To enhance the generalization process and increase the number of images in the validation set, six different data augmentation techniques [35] have been applied. The augmentation techniques used are image flipping, noise injection, gamma correction, Principal Component Analysis color augmentation, rotation, and scaling. Figure 7 provides the different augmentation operations applied to a leaf image and Table 4 presents the number of images before and after augmentation along with the number of images in the training and testing set respectively. Table 5 presents the data distribution of the field dataset in terms of training, validation, and testing. The dataset is partitioned into three where 70% is used for training, 20% is used for validation, and 10% is used for testing.

4.3 Hyperparameter tuning

The pre-trained DenseNet-121 architecture is altered by making minor changes before applying the transfer learning principle and after that, it takes its original structure. The significant challenge that arises here is to replace the old classifier with a novel one. This can be done by modifying the already existing hyperparameter value or adding new ones. The hyperparameters that are optimized in the proposed work are the number of neurons present in the fully connected layer, batch size, and learning rate. The additional hyperparameter added in this work is the drop-out layer rate. In this way, the M-HGSO algorithm optimizes three hyperparameters namely the number of neurons in the first fully connected layer, batch size, and newly added drop-out rate. This makes the search space of the M-HGSO algorithm three-dimensional and each gas particle indicates the combination of these three hyperparameters.

4.4 Training of DenseNet-121 architecture

To train the DenseNet-121architecture using the PlantVillage dataset, two techniques namely feature extraction and fine-tuning are used. The convolutional base of the DenseNet-121 architecture is kept unchanged during the feature extraction process. The classifier base is the one that is changed in this step using a novel classifier that fits the PlantVillage dataset. The novel classifier used in the feature extraction step comprises four layers namely the flatten layer, two fully connected layers, and a new dropout layer that divide the two fully connected layers. The first fully connected layer uses the ReLU as an activation function while the last fully connected layer uses the softmax function. The hyperparameters of these layers are determined using the M-HGSO algorithm. The novel classifier is trained for a certain number of iterations and enters the fine-tuning phase. In the fine-tuning phase, the last two blocks of the convolutional base are replaced with the novel classifier.

5 Experimental results and analysis

In this section, the results obtained using the proposed methodology are analyzed in detail along with the different performance metrics used.

5.1 Performance evaluation metrics

The proposed methodology is evaluated using different performance metrics namely accuracy (A), Precision (P), Recall (R), F1-Score ($F1$), and Confusion matrix. The accuracy mainly measures the ability of a machine learning classifier to classify the input image correctly. It identifies the number of accurately classified images out of all the images in the dataset. If an image is accurately classified as diseased or healthy means it is represented as a true

Table 3 Dataset description of the field dataset

Disease name	Image sample	Description	Symptom
Citrus Canker		This disease is caused by *Xanthomonas citri*. It makes the fruits unsightly and decreases their market value	Twig dieback, premature leaf, fruit drop, and blemished fruits
Citrus Scab		This disease develops in spring flush and survives in the infected leaves throughout overwintering lesions	Small semi-translucent dots like lesions and circular depression with a pink to the red center
Citrus Greasy Spot		It is caused by *Mycosphaerella citri* and it ruins the fruit crop and causes every leaf to fall	Black and yellow spots on the upper surface and pale orange to yellowish blister in the lower surface of the leaves cause a greasy appearance
Bacterial Canker in grapevine		It is caused by *Xylophilus ampelinus* bacteria and it mainly affects the plants while pruning	Abnormal colors and necrotic areas in leaves
Powdery mildew in Grapevine		This disease is mainly caused by the *Uncinula necator* fungus and it affects every green tissue present in the grapevine	Small white and grayish patches of fungal growth
Powdery Mildew in Melon		It is caused by the *Podosphaera xanthii* fungus which produces mycelium and spores on the outer surface	White fluffy mycelium present in both the upper and lower areas of the leaves
Mosiac in Melon		It is caused by the Cucumber Mosaic Virus which makes the plants severely stunted and the leaves are one size smaller than the rest	Foliages are covered in yellow mosaic, leaves of the plants are curled downwards and the size of the leaves are smaller
Smallpox in papaya		It is caused by *Asperisporium caricae* fungus and results in loss of fruits	Small furry looking yellow or brown spots in leaves
Mosaic Virus in papaya		It is caused by *potex* virus and it mainly forms in young leaves. The plant affected with the mosaic virus has a moderately stunted growth	Dark green blister-like patches and mild mosaic patterns are formed in leaves with slight deformation

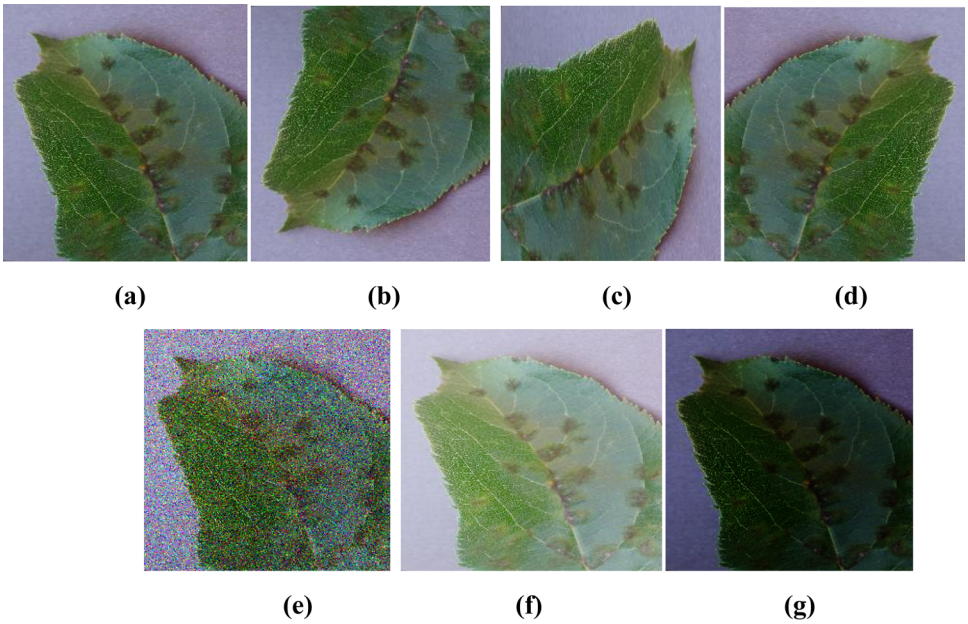


Fig. 7 Data augmentation procedure in the PlantVillage dataset. (**a**) Original image and (**b**–**g**) Augmented image

Table 4 Distribution of the plant Village dataset in terms of training, validation, and testing

Sl.No	Leaf class	Images before augmentation	Images after augmentation	Training dataset	Validation dataset	Testing dataset
1	Apple_Healthy	1645	1645	1184	329	132
2	Grape_Healthy	423	1000	720	200	80
3	Potato_Healthy	152	1000	720	200	80
4	Corn_Healthy	1162	1162	837	232	93
5	Apple_ scab	630	1000	720	200	80
6	Apple_blackrot	621	1000	720	200	80
7	Apple_rust	275	1000	720	200	80
8	Grape_Blackrot	1180	1180	850	236	94
9	Grape_Esca	1383	1383	995	277	111
10	Potato_Early Blight	1000	1000	720	200	80
11	Potato_Late Blight	1000	1000	720	200	80
12	Corn_Gray leaf spot	513	1000	720	200	80
13	Corn_Common rust	1192	1192	859	238	95
14	Corn_Leaf blight	985	1000	720	200	80
Total		12,161	15,562	11,205	3112	1245

positive (T_{pos}) or true negative (T_{neg}). If a diseased image is classified as healthy means it is indicated as false positive (F_{pos}) and if the healthy image is classified as diseased means it is indicated as a false negative (F_{neg}). The accuracy is mainly integrated with different performance metrics such as precision and recall to give a detailed view of classifying the leaf images. The error rate is the exact opposite of accuracy and it computes the number of incorrect predictions made by the model.

Precision is defined as the percentage of accurately identified positive image instances from all the predicted positive cases. The metric finds out how many images labeled as abnormal actually belong to the abnormal leaf disease class. The recall is defined as the percentage of accurately identified positive image instances to the actual positive predictions made. It identifies that from the overall number of disease classes how many images did the model label accurately. The precision metric can be useful when the cost associated with the number of false positives is

Table 5 Distribution of the field dataset in terms of training, validation, and testing

Sl.No	Leaf class	Total number of images in the dataset	Training dataset	Validation dataset	Testing dataset
1	Citrus Canker	950	665	190	95
2	Citrus Scab	874	612	175	87
3	Citrus Greasy Spot	1002	701	200	100
4	Bacterial Canker in grapevine	1254	878	251	125
5	Powdery mildew in Grapevine	1021	715	204	102
6	Powdery Mildew in Melon	958	671	192	96
7	Mosiac in Melon	969	678	194	97
8	Smallpox in papaya	1005	703	201	101
9	Mosaic Virus in papaya	1246	871	249	125
10	Citrus Healthy	1045	732	209	105
11	Papaya Healthy	1107	775	221	111
12	Grapevine healthy	1231	862	246	123
13	Melon healthy	1014	710	203	101
Total		13,676	9573	2735	1368

Table 6 Confusion Matrix Format

Predicted class values	Actual class values	
	Positive (1)	Negative (0)
Positive (1)	T_{POS}	F_{POS}
Negative (0)	F_{NEG}	T_{NEG}

high and the recall metric can be useful when the cost associated with the number of false negatives is high. The tradeoff between precision and recall can be identified using F1-Score and it is can also identify the unbalanced class distribution. The F1-score metric helps to determine the efficiency of the model when the number of false positives and false negatives are high. Equations (18–23) provides the different performance metrics discussed above.

$$A = \frac{T_{POS} + T_{NEG}}{T_{POS} + T_{NEG} + F_{POS} + F_{NEG}} \tag{18}$$

$$Error\_Rate = 1 - A \tag{19}$$

$$P = \frac{T_{POS}}{T_{POS} + F_{POS}} \tag{20}$$

$$R = \frac{T_{POS}}{T_{POS} + F_{NEG}} \tag{21}$$

Table 7 Parameter setting of the MHGSO optimized DenseNet-121 architecture

Parameter	Range
Number of Gases (Population)	30
Maximum number of iterations	15
Problem Dimension	3
β	1
η	1
Number of clusters	2–6
Training epochs of DenseNet121 architecture	10
Optimized hyperparameter value for batch size	8
Optimized hyperparameter value for the number of neurons in the first fully connected layer	120
Optimized hyperparameter value for dropout rate	0.1

Table 8 Performance evaluation of the different classes present in the PlantVillage dataset using the proposed methodology

Sl.No	Leaf class	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
1	Apple_Healthy	100	99	100	99
2	Grape_Healthy	100	100	100	100
3	Potato_Healthy	100	100	100	99
4	Corn_Healthy	100	100	98	97
5	Apple_ scab	98	99	98	98
6	Apple_blackrot	98	97	96	97
7	Apple_rust	97	98	99	98
8	Grape_Blackrot	97	98	98	97
9	Grape_Esca	98	98	98	98
10	Potato_Early Blight	98	98	99	98
11	Potato_Late Blight	99	99	99	98
12	Corn_Gray leaf spot	98	98	98	99
13	Corn_Common rust	100	100	100	99
14	Corn_Leaf blight	100	100	99	99
	Average	**98.7**	**98.6**	**98.7**	**98.2**
	Macro average	**98**	**98**	**98**	**98**
	Weighted average	**98**	**98**	**98**	**98**

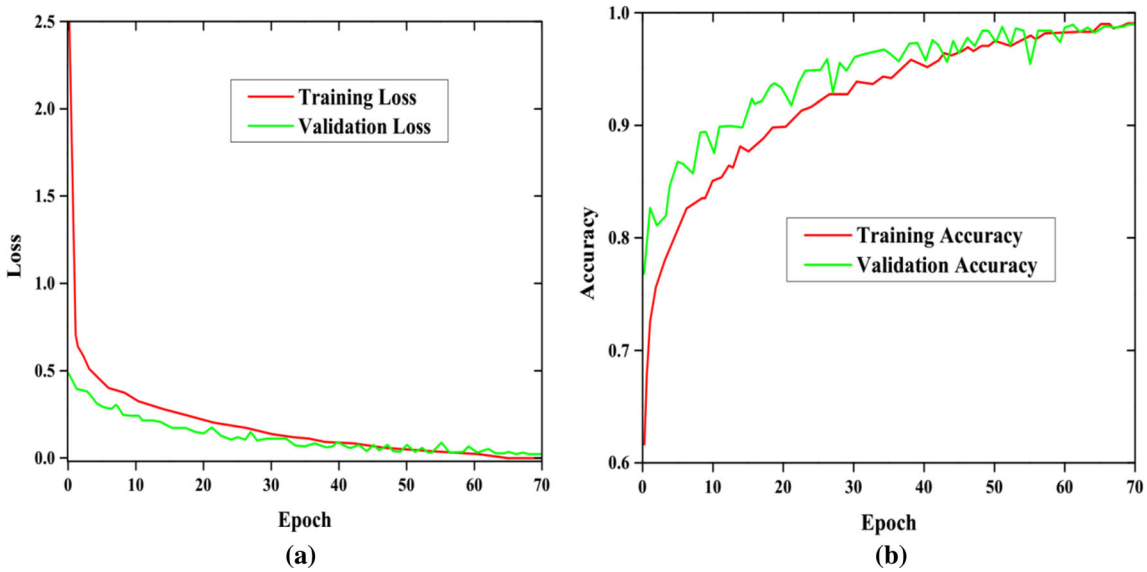


Fig. 8 Model's accuracy and loss obtained for different epochs. (**a**) Loss obtained for the training and validation set and (**b**) Accuracy achieved for the training and validation set

$$TPR = \frac{T_{POS}}{T_{POS} + F_{NEG}} \tag{22}$$

$$F1 = 2 * \frac{P * R}{P + R} \tag{23}$$

The Area Under The Receiver Operating Curve (AUC-ROC) is a performance metric that plots a graph between the True Positive Rate (TPR) and the False Positive Rate (FPR). Both the recall and the TPR value are the same. The confusion matrix shows the different classes predicted by the proposed model to test the system's efficiency. The confusion matrix format is presented in Table 6. The dimensions of the confusion matrix are mainly decided based on the number of classes to be classified by the proposed model. Since the proposed model utilizes 14 classes, a 14 × 14 dimension confusion matrix is used. It

Table 9 Confusion matrix for the validation dataset

Plant classes in PlantVillage dataset		Actual Class														Classification Accuracy of individual classes (%)
		P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	
Predicted class	P 1	**326**	2	0	1	0	0	0	0	0	0	0	0	0	0	**99**
	P 2	0	**200**	0	0	0	0	0	0	0	0	0	0	0	0	**100**
	P 3	0	1	**198**	1	0	0	0	0	0	0	0	0	0	0	**99**
	P4	0	0	0	**232**	0	0	0	0	0	0	0	0	0	0	**100**
	P5	0	0	0	0	**196**	0	0	1	1	0	2	0	0	0	**98**
	P6	0	0	0	0	0	**196**	0	0	0	0	0	0	0	0	**98**
	P 7	0	0	0	0	0	0	**194**	0	0	0	0	0	0	0	**97**
	P 8	0	0	0	0	0	0	0	**231**	0	0	0	0	0	0	**98**
	P 9	0	0	0	0	1	2	0	0	**271**	1	2	0	0	0	**98**
	P10	0	0	0	0	0	0	0	0	0	**200**	0	0	0	0	**100**
	P11	0	0	0	0	0	0	0	0	0	2	**198**	0	0	0	**99**
	P12	0	0	0	0	0	0	0	0	0	0	0	**196**	0	0	**98**
	P13	0	0	0	0	0	0	0	1	0	2	2	0	**233**	**0**	**98**
	P14	0	0	0	0	0	0	0	0	0	0	**0**	**0**	2	**198**	**99**

Table 10 Confusion matrix for the training dataset

Plant classes in PlantVillage dataset		Actual class														Classification accuracy of individual classes (%)
		P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	
Predicted class	P 1	**1184**	0	0	0	0	0	0	0	0	0	0	0	0	0	**100**
	P 2	0	**720**	0	0	0	0	0	0	0	0	0	0	0	0	**100**
	P 3	0	0	**720**	0	0	0	0	0	0	0	0	0	0	0	**100**
	P4	0	0	0	**837**	0	0	0	0	0	0	0	0	0	0	**100**
	P5	0	0	0	0	**706**	2	5	3	2	2	0	0	0	0	**98**
	P6	0	0	0	0	0	**706**	2	4	4	4	0	0	0	0	**98**
	P 7	0	0	0	0	6	5	**698**	5	5	1	0	0	0	0	**97**
	P 8	0	0	0	2	6	8	10	**824**	0	0	0	0	0	0	**97**
	P 9	0	0	0	0	0	10	5	5	**975**	0	0	0	0	0	**98**
	P10	0	0	0	0	4	4	0	3	0	**706**	3	0	0	0	**98**
	P11	0	0	0	0	0	0	0	3	2	**2**	**713**	0	0	0	**99**
	P12	0	0	0	0	0	0	3	0	3	0	0	**706**	3	5	**98**
	P13	0	0	0	0	0	0	0	0	0	0	0	0	**859**	0	**100**
	P14	0	0	0	0	0	0	0	0	0	0	0	0	0	**720**	**100**

gives a clear view of the correct and wrong predictions made by the model by mapping both the values. The actual classes are indicated using the rows and the predicted class is indicated using the columns. Each cell in the matrix represents the true positive, true negative, false positive, and false negative values based on the mapping of the proposed model.

5.2 M-HGSO parameter setup for hyperparameter selection

The M-HGSO algorithm sets the boundaries of the hyperparameters in the search space as follows. The initial values of the optimization parameters such as the batch size, dropout rate, and the number of neurons fall in a boundary of (1,64), (0.1–0.9), and (5,500) respectively. The parameter setting of the proposed methodology is

Table 11 Performance evaluation of the DenseNet-121 architecture and MHGSO algorithm by comparing it with LeNet and ABC algorithm

Metrics for comparison	Proposed MHGSO optimized DenseNet-121 architecture	ABC optimized DenseNet-121 architecture	LeNet [24]
Accuracy (%)	98.7%	94.6%	95.3%
MAP (%)	98%	94%	95%
MAR (%)	98%	94%	95%
MAF1-Score (%)	98%	94%	95%

presented in Table 7. The training epochs for the DenseNet-121 architecture are selected by experimenting with more than one value. By experimenting with it we observed that when the number of epochs exceeds 10, the training time for the M-HGSO takes exponential time. So, the number of epochs required to train the DenseNet121 architecture was set to 10. In this way, the loss of the validation set is controlled by the M-HGSO algorithm. After training the M-HGSO algorithm for 8 h, the optimal values for batch size, dropout rate, and the number of neurons are determined.

5.3 Training DenseNet 121 architecture using optimized hyperparameters

In this section, the training of the DenseNet-121 architecture with the optimal parameters selected by the M-HGSO algorithm is used. The training was done in the training dataset and the evaluation was done in the testing dataset for a total number of k epochs. To obtain the best value for k, the experiments were conducted for different numbers of times. The DenseNet obtained the best results on the validation dataset at the 32nd epoch for the feature extraction phase. The best result for the fine-tuning phase was achieved at the 40th epoch and after that, there was no improvement at all. Hence, the value of k is taken as 32 for the feature extraction stage and 40 for the fine-tuning phase. After no improvement is noticed after the eighth iteration, the training process is terminated before the repetition of k-epochs to prevent overfitting. The training process is terminated using an early stopping strategy [37]. For the PlantVillage dataset, the DenseNet 121 architecture is compiled using a multi-class weighted cross entropy technique [38]. For the feature extraction phase, the adam optimization algorithm with a constant learning rate of 2e-5 is used. For the fine-tuning phase, a step decay schedule is used. The initial learning rate of the model is taken as 1e-5 and the value drops by 0.5 for every training epochs. The low learning rate used in the fine-tuning model ensures that only a slight change occurs during this process such that the crucial features obtained from the feature extraction phase are not lost.

5.4 Computational complexity of the M-HGSO algorithm

The proposed M-HGSO algorithm has the same computational complexity as the HGSO algorithm. The mutation operation added doesn't increase the complexity any higher. The complexity is mainly based on the population size (M), the number of dimensions (D), the individual rank of the gas particles, and the total number of iterations (i_{max}). The computational complexity of the M-HGSO algorithm is formulated as $O(i_{max} \times M \times D) \times O(obj)$.

Table 12 Comparative analysis with the state-of-art techniques

Source	Name of the model used	Number of classes used	Accuracy	Precision	Recall
Miaomiao Ji et al. [7]	UnitedModel based multiple CNN	3	98.57%	98.35%	98.65%
Abdul Waheed et al. [8]	Dense-CNN	3	98.06%	97.95%	98.14%
Chongke Bi et al. [23]	MobileNet	3	73.50%	73.25%	73.65%
Ramar Ahila Priyadharshini et al. [24]	LeNet based CNN architecture	3	97.89%	97.53%	97.92%
Mohit Agarwal et al. [11]	Lightweight CNN	5	98.4%	98%	98.41%
–	Proposed MHGSO optimized DenseNet-121 architecture (PlantVillage dataset)	14	98.7%	98.64%	98.78%
–	Proposed MHGSO optimized DenseNet-121 architecture (Field dataset)	13	98.81%	98.60%	98.75%

Table 13 Comparison of the proposed method with the public (PlantVillage) and field dataset

Source	Name of the model used	Accuracy for PlantVillage dataset	Accuracy for field dataset
Miaomiao Ji et al. [7]	UnitedModel based multiple CNN	92%	85.57%
Abdul Waheed et al. [8]	Dense-CNN	90%	82.06%
Chongke Bi et al. [23]	MobileNet	71.201%	71.50%
Ramar Ahila Priyadharshini et al. [24]	LeNet based CNN architecture	92%	87.15%
Mohit Agarwal et al. [11]	Lightweight CNN	91%	88.4%
–	Proposed MHGSO optimized DenseNet-121 architecture	98.7%	98.8%

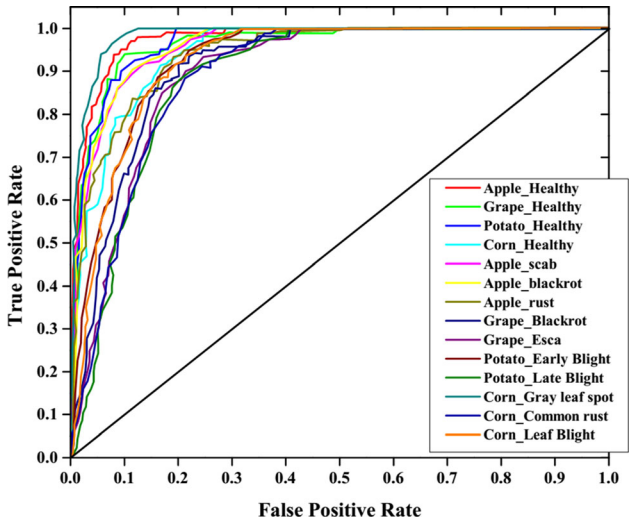


Fig. 9 ROC curve for the training dataset (PlantVillage)

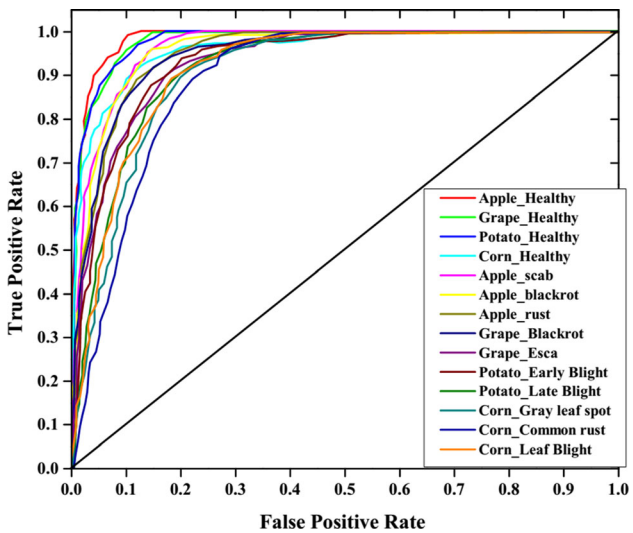


Fig. 10 ROC curve for the testing dataset (PlantVillage)

The parameter O(*obj*) represents the complexity associated with the objective function.

5.5 Performance evaluation of the proposed approach

The performance of MHGSO optimized DenseNet-121architecture is evaluated in this section using different performance metrics such as accuracy, Error Rate, Precision, Recall, F1-Score, and True Positive Rate (TPR). The MHGSO optimized DenseNet-121 architecture achieved an accuracy of 98.71% in the PlantVillage dataset for the 14 classes. The average precision, recall, and F1-score for the 14 classes were 98.6%, 98.7%, and 98.2% respectively. The average, macro average, and the weighted average for the different performance metrics such as accuracy, precision, and recall are presented in Table 8.

The proposed MHGSO optimized DenseNet-121architecture's accuracy and loss are evaluated in terms of both the training and validation dataset as shown in Fig. 8a, b.

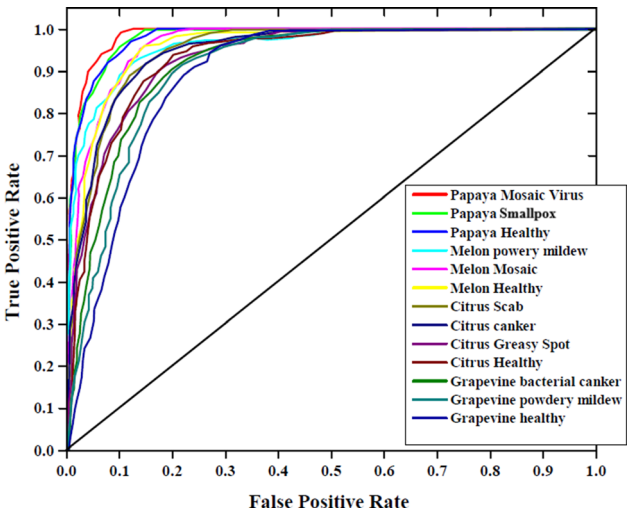


Fig. 11 ROC curve for the field dataset when evaluated with the training dataset

Table 14 Analysis in terms of computational time

Number of runs	Techniques used	Training time (in seconds)	Testing time (in seconds)
1	Dense CNN [8]	4.3251	1.5245
	MobileNet [6]	3.9572	1.5478
	LeNet based CNN architecture [24]	4.0214	1.5147
	Proposed MHGSO optimized DenseNet-121 architecture	0.5145	0.1054
2	Dense CNN [8]	4.2658	1.4215
	MobileNet [6]	4.0154	1.2145
	LeNet based CNN architecture [24]	4.1215	1.4178
	Proposed MHGSO optimized DenseNet-121 architecture	0.5872	0.1854
3	Dense CNN [8]	4.4568	1.6548
	MobileNet [6]	4.3556	1.3578
	LeNet based CNN architecture [24]	4.5145	1.4587
	Proposed MHGSO optimized DenseNet-121 architecture	0.6324	0.1185

The accuracy is computed after the parameter of the optimized DenseNet-121 architecture is trained and fixed. The test images are given as the input to the MHGSO optimized DenseNet-121 architecture and the error rate is compared with the actual class. The model's behavior after each iteration during optimization is computed using the loss value (negative likelihood). The robustness and convergence capacity of the proposed model can be verified via the accuracy and loss values obtained.

The confusion matrix helps to identify the number of leaf images incorrectly classified by the proposed technique. Tables 9 and 10 presents the confusion matrix for both the validation and training datasets respectively. The text highlighted in bold represents the image samples that were identified accurately while the misclassifications of each class are represented using normal text. The classes such as Apple scab, Apple black rot, cedar apple rust, grape black rot, grape esca, potato early blight, potato late blight, corn gray leaf spot, corn common rust, and corn leaf blight are represented as P1-P14 in Tables 9 and 10. From both the confusion matrix, one can observe that the misclassification rate is really less which shows the model's ability to classify the leaf diseases with few errors.

To evaluate the effectiveness of the proposed MHGSO algorithm in identifying the optimal hyperparameters of the DenseNet-121 architecture, it is being compared with the ABC optimization algorithm [11]. To ensure a fair comparison between both the algorithms, the parameters of the ABC algorithm have been set as the same as the MHGSO algorithm shown in Table 7. After the training process of the ABC algorithm is completed, the batch size, dropout rate, and the number of neurons in the fully connected layer of the DenseNet are set as 7, 230, and 0.3 respectively. Based on the comparative results obtained using the test set it is clear that the MHGSO algorithm is efficient in selecting the optimal values for the DenseNet-121 hyperparameters to achieve a higher classification accuracy for classifying the different plant disease classes. The MHGSO optimized DenseNet architecture achieved an accuracy value of 98.7% whereas the ABC optimized DenseNet architecture achieved an accuracy of 94.6%. The comparison results are shown in Table 11, where the MAP, MAR, and MAF1-Score represent the macro average precision, macro average recall, and macro average F1-score respectively. The MAP, MAR, and MAF1-score of the ABC optimized DenseNet-121 architecture was 94%.

To evaluate the entire performance of the MHGSO optimized DenseNet-121 architecture, it is compared with the LeNet architecture [24] where the hyperparameters are selected randomly using manual search. Here the values of the dropout rate, number of neurons, and batch size were chosen as 0.5,240 and 15. The comparison results shown in Table 11 indicate that the performance of the proposed methodology is higher than the LeNet architecture. The accuracy of the LeNet architecture achieved for the test set was 95.3%. The proposed architecture is also compared with the state-of-art techniques in terms of different performance metrics and the number of classes used as shown in Table 12. Models such as UnitedModel based multiple CNN [7], Dense-CNN [8], LeNet [24], and Lightweight CNN [11] provide higher classification accuracy but are only capable of discriminating from three to five classes. The MobileNet [23] architecture provides a low accuracy value of 73.65% when compared to the remaining techniques. The proposed method offers a classification accuracy of 98.7% and is capable of recognizing 14 different leaf classes from a relatively large number of samples, illustrating the superiority of this scheme.

The proposed approach's accuracy is stable when evaluated with a field dataset with a complex background. The

proposed approach offers accuracy, precision, and recall scores of 98.81%, 98.60%, and 98.75% for 13 classes. Techniques such as UnitedModel based multiple CNN [7], Dense-CNN [8], MobileNet [23], LeNet based CNN architecture [24], and Lightweight CNN [11] provide higher accuracy only for 3 to 5 classes and are primarily used to classify disease classes for the same crop. As a result, these models are tested using the plant Village and Field datasets for a total of 14 and 13 classes, respectively, and the results are shown in Table 13. When evaluated with more than 10 classes, the accuracy of the conventional techniques decreases for both datasets. For the field dataset with sophisticated backgrounds, the accuracy of the UnitedModel based multiple CNN, Dense-CNN, MobileNet, LeNet based CNN architecture, and Lightweight CNN architecture is 85.57%, 82.06%, 71.50%, 87.15%, and 88.4%, respectively, and the accuracy is relatively low for their accuracies obtained in the Plant Village dataset. The proposed model offers a higher accuracy for both datasets as shown in Table 13.

The ROC curves indicate the reliability of the model from 0 to 1. The output value near one indicates the model can differentiate more than one class with higher accuracy. If the value is less than or equal to 0.5, this means that the model suffers from poor accuracy when distinguishing classes. The multiple curves in Figs. 9 and 10 represent the different leaf disease classes classified by the proposed model using the PlantVillage dataset. The diagonal in the middle represents that the curves should not go below it. Figures 9 and 10 present the ROC curves for the training and testing dataset. The healthy leaf classes are accurately classified for the four crop variants as shown in Figs. 9 and 10. In both the ROC graphs, the value of every ROC curve reaches 1, which shows that the proposed methodology offers higher performance. Figure 11 presents the ROC curve for the field dataset (training). Since the output is nearly equal to one for most of the classes, the proposed model offers an increased accuracy even for images with complex backgrounds.

5.6 Computational time

The performance of the proposed algorithm is further evaluated using the computational time by comparing it with different techniques such as Dense CNN [8], MobileNet [6], and LeNet based CNN [24]. Every algorithm was run consecutively for three runs. The results obtained for the three techniques in terms of computational time are presented in Table 14. In the first run, the time taken by the proposed MHGSO optimized DenseNet-121 architecture is 0.5145 s which is relatively low than the Dense CNN, MobileNet, and LeNet based CNN techniques. The training time of the proposed MHGSO optimized DenseNet-121 architecture is eight times lower than Dense CNN and LeNet based CNN techniques and seven times lower than that of MobileNet. The proposed MHGSO optimized DenseNet-121 architecture requires a testing time of 0.1185 s which is relatively lower than the testing time taken by the Dense CNN, MobileNet, and LeNet based CNN techniques by a factor of one to two. The results demonstrate that the performance of the proposed MHGSO optimized DenseNet-121 architecture is superior to the state-of-art techniques.

6 Conclusion

This paper proposes an MHGSO optimized DenseNet-121 architecture for plant leaf disease classification for the images obtained from the PlantVillage and field dataset. The three main stages of the proposed architecture are data preprocessing, hyperparameter tuning, and training the DenseNet-121 architecture. In the data processing stage, the images in the PlantVillage dataset is subjected to augmentation and divided into testing and training dataset. The hyperparameter tuning phase modifies the existing hyperparameter values of the DenseNet-121 architecture using the MHGSO algorithm by applying the transfer learning technique. The learning rate, batch size, and the number of neurons in the fully connected layer are selected for optimization. The DenseNet-121 architecture is trained using the fine-tuning and feature extraction phase for classification. This work also includes the concept of cloud computing to retrieve and process the image obtained from the farmers online using the proposed technique. The hyperparameter optimization along with the transfer learning approach used in this proposed work yields a classification accuracy of 98.7%. The proposed approach was compared using different performance evaluation metrics such as accuracy, recall, confusion matrix, ROC curve, and F1-Score. The proposed MHGSO optimized DenseNet-121 architecture offers accuracy, precision, and recall scores of 98.81%, 98.60%, and 98.75% when evaluated with the field dataset with complex backgrounds. As a possible future work, we plan to increase the plant diversity and the number of disease classes and use big data for processing these images. In addition, the disease severity of the diseased leaf image class can also be identified by designing a novel optimized model with less computation time.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Madhusudhan L (2015) Agriculture role on Indian economy. Bus Econ J 6(4):1
2. Roy D, Thorat A (2008) Success in high value horticultural export markets for the small farmers: the case of Mahagrapes in India. World Dev 36(10):1874–1890
3. Vega GL (2015) Image-based detection and classification of allergenic pollen. PhD diss., Université de Bourgogne
4. Athiraja A, Vijayakumar P (2020) Banana disease diagnosis using computer vision and machine learning methods. J Ambient Intell Hum Comput 2:1–20
5. Thangaraj R, Anandamurugan S, Kaliappan VK (2020) Automated tomato leaf disease classification using transfer learning-based deep convolution neural network. J Plant Dis Protect 2:1–14
6. Argüeso D, Picon A, Irusta U, Medela A, San-Emeterio MG, Bereciartua A, Alvarez-Gila A (2020) Few-shot learning approach for plant disease classification using images taken in the field. Comput Electron Agric 175:105542
7. Ji M, Zhang L, Qiufeng Wu (2020) Automatic grape leaf diseases identification via UnitedModel based on multiple convolutional neural networks. Inf Process Agric 7(3):418–426
8. Waheed A, Goyal M, Gupta D, Khanna A, Hassanien AE, Pandey HM (2020) An optimized dense convolutional neural network model for disease recognition and classification in corn leaf. Comput Electron Agric 175:105456
9. Sundararaj V (2016) An efficient threshold prediction scheme for wavelet based ECG signal noise reduction using variable step size firefly algorithm. Int J Intell Eng Syst 9(3):117–126
10. Barbedo JG (2018) Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification. Comput Electron Agric 153:46–53
11. Andrushia AD, Trephena Patricia A (2020) Artificial bee colony optimization (ABC) for grape leaves disease detection. Evol Syst 11(1):105–117
12. Zhao B, Huang Bo, Zhong Y (2017) Transfer learning with fully pretrained deep convolution networks for land-use classification. IEEE Geosci Remote Sens Lett 14(9):1436–1440
13. Jose J, Gautam N, Tiwari M, Tiwari T, Suresh A, Sundararaj V, Rejeesh MR (2021) An image quality enhancement scheme employing adolescent identity search algorithm in the NSST domain for multimodal medical image fusion. Biomed Signal Process Control 66:102480
14. Vallabhajosyula S, Sistla V, Kolli VKK (2021) Transfer learning-based deep ensemble neural network for plant leaf disease detection. J Plant Dis Protect 2:1–14
15. Atila Ü, Uçar M, Akyol K, Uçar E (2021) Plant leaf disease classification using EfficientNet deep learning model. Ecol Inf 61:101182
16. Lu J, Tan L, Jiang H (2021) Review on convolutional neural network (CNN) applied to plant leaf disease classification. Agriculture 11(8):707
17. Nandhini S, Ashokkumar K (2021) Improved crossover based monarch butterfly optimization for tomato leaf disease classification using convolutional neural network. Multimed Tools Appl 80:18583–18610. https://doi.org/10.1007/s11042-021-10599-4
18. Ezzat, Dalia, Aboul Ella Hassanien, and Hassan Aboul Ella. "An optimized deep learning architecture for the diagnosis of COVID-19 disease based on gravitational search optimization." *Applied Soft Computing* (2020): 106742.
19. Darwish A, Ezzat D, Hassanien AE (2020) An optimized model based on convolutional neural networks and orthogonal learning particle swarm optimization algorithm for plant diseases diagnosis. Swarm Evol Comput 52:100616
20. Hinz T, Navarro-Guerrero N, Magg S, Wermter S (2018) Speeding up the hyperparameter optimization of deep convolutional neural networks. Int J Comput Intell Appl 17(02):1850008
21. Cabada RZ, Rangel HR, Estrada MLB, Lopez HMC (2020) Hyperparameter optimization in CNN for learning-centered emotion recognition for intelligent tutoring systems. Soft Comput 24(10):7593–7602
22. Aszemi NM, Dominic PDD (2019) Hyperparameter optimization in convolutional neural network using genetic algorithms. Int J Adv Comput Sci Appl 10(6):269–278
23. Bi C, Wang J, Duan Y, Baofeng Fu, Kang J-R, Shi Y (2020) Mobilenet based apple leaf diseases identification. Mobile Netw Appl 2:1–9
24. Priyadharshini RA, Selvaraj A, Madakannu A, Annamalai M (2019) Maize leaf disease classification using deep convolutional neural networks. Neural Comput Appl 31(12):8887–8895
25. Agarwal M, Gupta SK, Biswas KK (2020) Development of Efficient CNN model for Tomato crop disease identification. Sustain Comput Inf Syst 28:100407
26. Cristin R, Santhosh Kumar B, Priya C, Karthick K (2020) Deep neural network based Rider-Cuckoo Search Algorithm for plant disease detection. Artif Intell Rev 2:1–26
27. Yogeshwari M, Thailambal G (2021) Automatic feature extraction and detection of plant leaf disease using GLCM features and convolutional neural networks. Mater Today Proc
28. Appalanaidu MV, Kumaravelan G (2021) Plant leaf disease detection and classification using machine learning approaches: a review. Innov Comput Sci Eng 2:515–525
29. CS231n: Convolutional Neural Networks for Visual Recognition. Stanford University CS231n: Convolutional Neural Networks for Visual Recognition. [Online]. Available: http://cs231n.stanford.edu/. [Accessed: 22-Jan-2021].
30. Zare S, Moosa A (2020) Simultaneous fault diagnosis of wind turbine using multichannel convolutional neural networks. ISA Trans
31. Kumar A, Gandhi CP, Zhou Y, Kumar R, Xiang J (2020) Improved deep convolution neural network (CNN) for the identification of defects in the centrifugal pump using acoustic images. Appl Acous 167:107399
32. Hashim FA, Houssein EH, Mabrouk MS, Al-Atabany W, Mirjalili S (2019) Henry gas solubility optimization: A novel physics-based algorithm. Futur Gener Comput Syst 101:646–667
33. Sander R (2015) Compilation of Henry's law constants (version 4.0) for water as solvent. Atmos Chem Phys 15(8):4399–4981
34. Hughes D, Marcel S. An open access repository of images on plant health to enable the development of mobile disease diagnostics. arXiv preprint arXiv:1511.08060 (2015).
35. Geetharamani G, Pandian A (2019) Identification of plant leaf diseases using a nine-layer deep convolutional neural network. Comput Electr Eng 76:323–338
36. Repositório Digipathos: Base De Imagens De Sintomas De Doenças De Plantas PDDB, www.digipathos-rep.cnptia.embrapa.br/jspui/handle/123456789/2.
37. Raskutti G, Wainwright MJ, Bin Yu (2014) Early stopping and non-parametric regression: an optimal data-dependent stopping rule. J Mach Learn Res 15(1):335–366
38. Akil M, Saouli R, Kachouri R (2020) Fully automatic brain tumor segmentation with deep learning-based selective attention using overlapping patches and multi-class weighted cross-entropy. Med Image Anal 63:101692