INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

Multi-Class Sunflower Disease Detection by Integrating Enhanced Jellyfish Search Algorithm

T Thilagavathi[1]*, **L Arockiam**[2]

1 Research Scholar, Department of Computer Science, St. Joseph's College (Autonomous), Affiliated to Bharathidasan University, Tiruchirappalli, 620002, Tamil Nadu, India
2 Associate Professor, Department of Computer Science, St. Joseph's College (Autonomous), Affiliated to Bharathidasan University, Tiruchirappalli, 620002, Tamil Nadu, India

OPEN ACCESS

Received: 06-09-2024
Accepted: 12-11-2024
Published: 04-12-2024

Citation: Thilagavathi T, Arockiam L (2024) Multi-Class Sunflower Disease Detection by Integrating Enhanced Jellyfish Search Algorithm. Indian Journal of Science and Technology 17(44): 4633-4645. https://doi.org/10.17485/IJST/v17i44.2874

* **Corresponding author**.

thilagamca96@gmail.com

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objectives: This research aims to classify diseases observed in sunflower flowers and leaves using deep learning algorithms. The goal is to enhance agricultural output by addressing the significant issue of plant diseases, which negatively affect the security of the food supply. **Methods:** The study utilizes a feature selection approach to identify multi-class sunflower diseases. The Enhanced Jellyfish Search (EJFSOA) algorithm is improved in three ways: (i) optimization capabilities and convergence speed are enhanced through sine and cosine learning variables during Type B motion in the swarm, (ii) a local escape operator is added to avoid the local optimization trap, improving the exploitability of the JS algorithm, and (iii) opposition-based learning and quasi-opposition learning strengthen the population distribution. A deep learning model, Multi-class Sunflower Network (MCSFNet), combines ResNets with TensorFlow for feature extraction, multi-label classification, and a multi-spectral stacking module. **Findings:** The proposed model is evaluated using several criteria, including accuracy, sensitivity, specificity, and F1-score. On a publicly available dataset, the algorithm achieves an average accuracy of 99.56%, outperforming state-of-the-art approaches with accuracies of 98.34% and 97.39%. **Novelty:** This work introduces improvements to the EJFSOA algorithm, significantly enhancing the precision and convergence speed of sunflower disease classification models. Combining deep learning with an optimized feature selection method leads to superior results in multi-class classification tasks.

Keywords: Sunflower Leaves; Disease Detection; Enhanced jellyfish search; Local escape operator; Multiclass Sunflower Network; Residual Networks

1 Introduction

The sunflower, or Helianthus annuus, has seeds that are not only edible but also contain oil [(1)]. In addition, the flowers provide a yellow dye and the leaves yield hay. Leaf scars, Septoria leaf spot, Verticillium wilt, downy mildew, and Phom blight are some of the

diseases that harm sunflower crops[2]. Machine learning (ML) techniques are being employed to enhance global food security by making plant disease detection faster and more accurate[3]. This study employs a deep learning classifier based on feature selection to detect sunflower diseases across multiple classes. By using a local escape operator, a learning method, sine and cosine variables, and the EJFSOA model, the most crucial features are chosen. Optimization capabilities are improved, and the convergence rate is accelerated by introducing sine and cosine learning parameters, allowing jellyfish to learn from the optimal individual's position while moving in Type B form within the group. To improve the JS method's exploitability, a local escape operator is inserted, enabling the algorithm to avoid local optimization traps. The algorithm's accuracy, convergence speed, and solution quality are enhanced by combining opposition-based learning with quasi-opposition learning strategies, leading to a more diversified distribution of candidate individuals. The classification is performed using the MCSFNet model, which combines ResNets with TensorFlow for feature extraction, a multi-label classification module, and a multi-spectral stacking module[4]. By increasing the number of channels, this module ensures the transfer of more detailed gen, offering an extended feature representation space, and also collects global info as base-level features from feature maps. To further improve classification accuracy, the input and fully linked layers of the network are designed to make the network more resilient to changes and noise in the input images. Because it is specifically designed to output a likelihood for each class independently, a multi-label function is ideal for applications that require the classification of several labels.

Recent works on sunflower disease classification have made significant progress, but several gaps remain. Existing studies have focused on comparing different deep learning architectures and achieved high accuracies; however, most did not incorporate advanced optimization techniques or feature selection methods, which could enhance model performance further[5]. Some research relied on limited datasets, lacking the ability to generalize results to larger and more complex environments. While lightweight models have been developed to improve efficiency, they often neglect advanced optimization strategies to refine feature selection and enhance classification accuracy across diverse conditions. These gaps highlight the need for combining deep learning with sophisticated feature selection and optimization techniques to achieve better accuracy, generalization, and efficiency in practical agricultural applications.

2 Methodology

The workflow is presented in Figure 1.

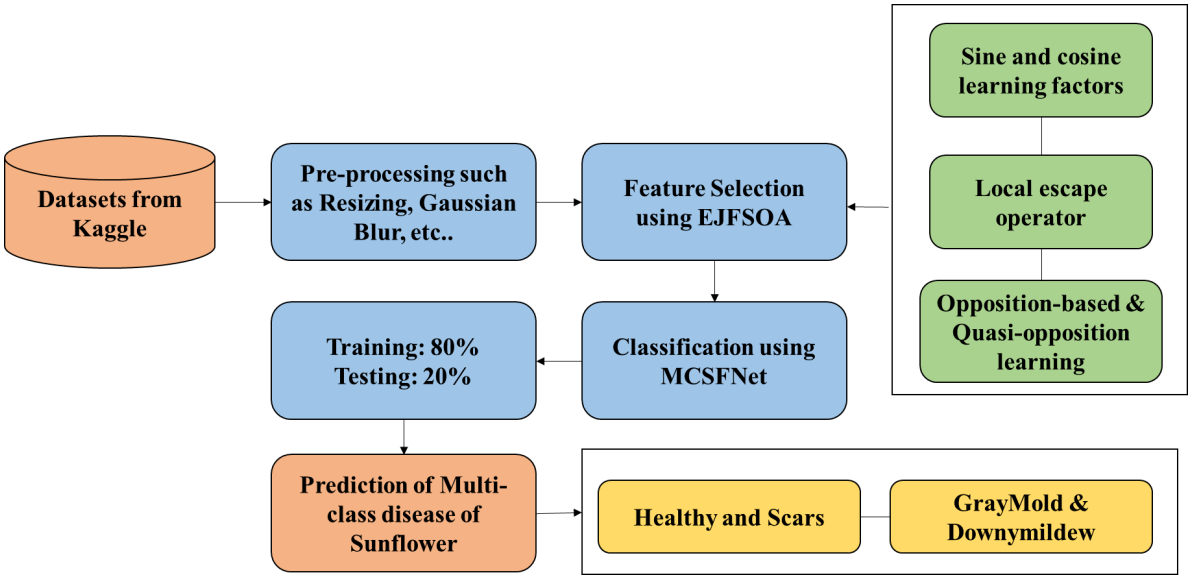

Fig 1. Workflow of projected model

An entire 467 pixels make up the original image. In all, there are 1668 augmented images. The augmentation is done by the data set's producers. Two types of general enlargement procedures are included in augmentation operations: location and color alteration. A standard resolution of 512 by 512 pixels is used throughout all of the images. There were four distinct picture categories in the dataset; the training set made use of 80% of the photographs, while the test set made use of 20%.

To know that training the deep learning model requires a significant amount of data, so should increase the number of photos. So, as the amount of data grows, so does the performance of the model[6]. Researchers used two broad augmentation techniques—location and color change—during the study's data augmentation phase. To utilize random rotation, random horizontal flip, and random approaches for increasing by position. Just brightness is applied throughout the color improvement process. The generalized model mentions the limitations of using smaller datasets. However, no additional large-scale experiments or testing on diverse datasets are reported, indicating a gap in exploring generalizability to broader, more varied agricultural conditions. There is limited information on testing with noisy data. The training dataset is enlarged by utilizing these augmentation strategies, giving the model access to a bigger and more varied collection of samples. The deep learning model's ability to generalize and perform better is enhanced by the greater variety of images. Figure 2 displays the dataset's sample distribution.

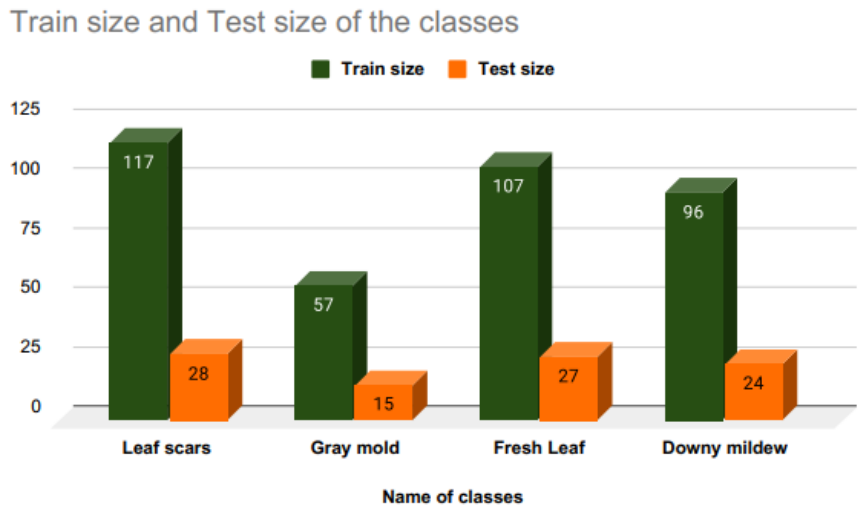

Fig 2. Number of samples for different classes

2.1 Data Preprocessing

The following is a synopsis of the various preprocessing methods used in this study. To acquire the pre-processed images after applying the procedures to all the photographs.

- Image Resizing: The photos are now 150 × 150 pixels in size, down from 512 × 512.
- Color Conversion: By default, OpenCV uses read to read images in BGR format. Transforming from BGR to RGB and back again is possible with the help of the cvtColor function[7].
- To smooth out an image, the GaussianBlur function in OpenCV takes the average of the pixels around each one and uses it to replace its pixel value, resulting in image blurring. It is a typical pre-processing step for computer vision tasks like edge identification since it decreases picture noise and detail.
- Image Denoising: fastNl Means Denoising Colored OpenCV for denoising colored images. It is based on the non-local means denoising algorithm, to effectively remove noise from images while preserving the details. It is fast and to be used for real-time applications[8]. In the study Some robustness in handling noisy inputs through its multi-layer classification module. However, specific performance metrics on noisy datasets are not provided, limiting insight into the model's performance under real-world conditions of data quality.
- Scaling and Converting: One OpenCV function that may be used to scale and determine the absolute values of a matrix or array is convertScaleAbs. In image processing besides computer vision tasks, it is commonly used to modify picture brightness, contrast, and other attributes by transforming the outcome of mathematical operations into a visually displayable representation. For brightness control, use a = 1.00, and for contrast adjustment, use b = 0.

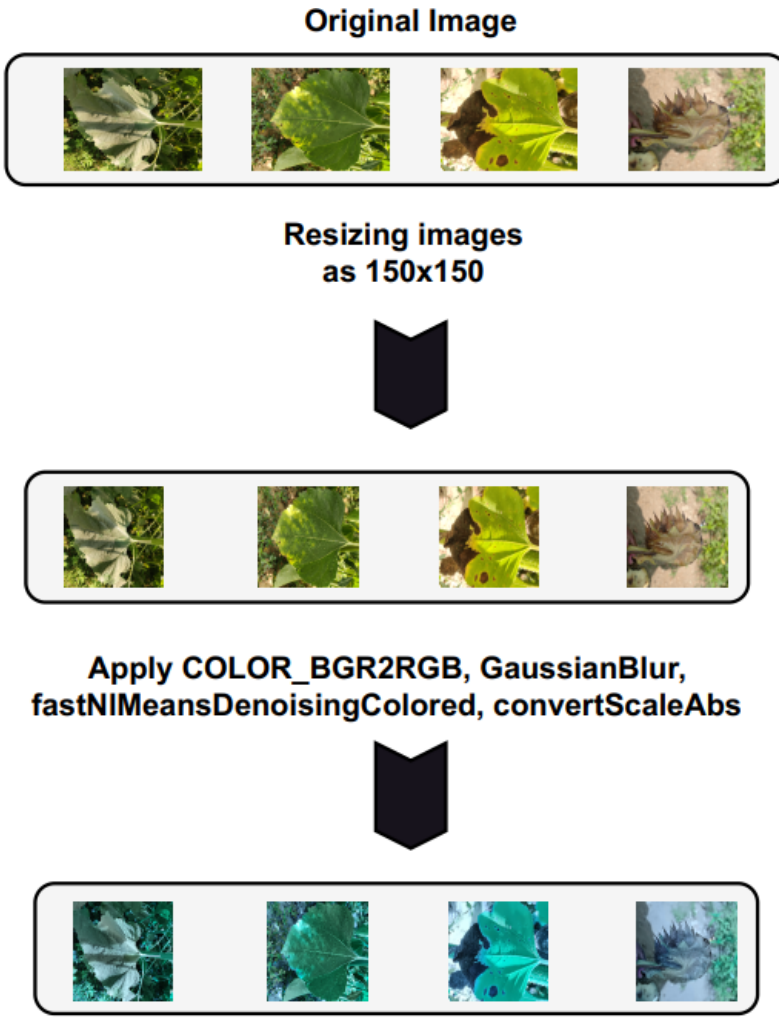

Fig 3. The pre-processed images from the original

2.2 Enhanced Jellyfish Search Procedure

The JS algorithm has the potential to boost solution quality, speed up convergence, and increase algorithm precision: (i) Jellyfish learn from the best individual simultaneously while moving in Type B motion, which improves the quality of the solution and speeds up convergence, by adding learning variables. (ii) The JS algorithm's utilization is enhanced since the local escape operator is included, which stops the algorithm from becoming trapped at a local optimal value. (iii) The solution's quality and the JS algorithm's speed besides accuracy are improved by smearing an opposition-based learning strategy to upsurge candidate populations. In the next iteration, the best folks from both the current and new solutions are executed, further improving the algorithm's performance.

A) Sine and Cosine Learning Factors

As part of the JS procedure's exploration phase, when a swarm of jellyfish is moving in Type B motion, the rationalized site of each jellyfish is only associated with another randomly picked jellyfish. In essence, the present population of jellyfish is blind and does not exchange knowledge effectively; as a result, the jellyfish learn at random from one another. In addition to slowing convergence, this procedure may cause the algorithm to veer from the path of the best possible candidate solution[9]. The jellyfish was made to learn from people and the best person in the search range by introducing the sine and cosine learning variables, w1 and w2, respectively. Through faster convergence and improved candidate solution quality during exploration,

this technique finds the optimal position more quickly.

$$\omega_1 = 2.sin\left[\left(1-\frac{t}{T}\right).\pi/2\right] \tag{1}$$

$$\omega_2 = 2.os\left[\left(1-\frac{t}{T}\right).\pi/2\right] \tag{2}$$

In Type B crusade, Equation (3) defines the location apprise manner of jellyfish:

$$P_i\left(t+1\right) = \omega_1.\left(P_i\left(t\right)+\overrightarrow{step}\right)+\omega_2\left(P^*-P_i(t)\right) \tag{3}$$

Where $\overrightarrow{step}$ This can seen from original jellyfish optimization algorithms.

Jellyfish, according to the original JS procedure, learn at random from each individual since it uses a stochastic learning technique. Because the learned jellyfish individuals have low fitness values, the convergence speed will be constrained.[10] Thus, the jellyfish algorithm learns from random resolutions and follows the optimal range to the sine and cosine learning factors included in the JS procedure. This rapidly improves solution quality and speeds convergence speed.

B) Local Escape Operator

Assessing and balancing an algorithm's capabilities is fundamental to swarm intelligence algorithms. While the JS procedure's capacity for local exploration is enhanced with the additional variables, its capacity for global exploitation is diminished. To discover new areas and improve the exploitation capability, the local escape operator (LEO) searches locally using a gradient-based optimizer (GBO). Consequently, it is utilized when jellyfish are in the process of swimming with the flow of the ocean. Notably, the position of the candidate $P_i\left(t+1\right)$. The algorithm is used to bypass solutions that are locally optimal because of this. This means it's to find the best solution on a global scale by including a wider range of candidates. Put simply, it prevents the algorithm from falling into the pitfall of local optimization.

Optimal individual P^* besides two solutions chosen at random are examples of several possible solutions. $P1_i(t)$ and $P2_i(t)$, two randomly designated candidate solutions $P_{r1}(t)$ and $P_{r2}(t)$, a novel candidate position $P_k(t)$, the LEO key $P_{LEO}(t)$ of the current solution $P_i(t\ +\ 1)$, besides the produced key is to probe the region surrounding the best answer. For a detailed explanation, refer to Equation (4) in addition to Equation (5):

If $rand < 0.5$

$$P_{LEO}\left(t\right) = P_i\left(t+1\right)+f_1\left(u_1P*-u_2P_k(t)\right)+f_2\rho_1\left(u_3\left(P2_i\left(t\right)-P1_i\left(t\right)\right)u_2\left(P_{r1}\left(t\right)-P_{r2}\right)\right)/2 \tag{4}$$

$$P_i\left(t+1\right) = P_{LEO}(t)$$

Else

$$P_{LEO}\left(t\right) = P^*+f_1\left(u_1P^*-u_2P_k(t)\right)+f_2\rho_1\left(u_3\left(P2_i\left(t\right)-P1_i(t)\right)\right)+u_2\left(P_{r1}\left(t\right)-P_{r1}(t)\right)/2 \tag{5}$$

$$P_i(t+1) = P_{LEO}(t)$$

END

where f_1 is any sum homogeneously scattered in $[-1,\ 1],\ f_2 \sim N\left(0,\ 1\right).u1,\ u2$, besides u3 are three random statistics with formulae:

$$u_1 = L_1\times 2\times R_1+\left(1-L_1\right) \tag{6}$$

$$u_2 = L_1\times R_2+\left(1-L_1\right) \tag{7}$$

$$u_3 = L_1 \times R_3 + (1 - L_1) \tag{8}$$

where L_1 is a binary parameter, besides L_1 = 0 or 1. If μ_1 < 0.5, L_1 = 1, else, L1 = 0, and μ_1 is defined as an arbitrary sum among 0 besides 1. In adding, ρ_1 is the adaptive coefficient; $R1 = rand(0, 1)$, $R2 = rand(0, 1)$, besides R3 = rand (0, 1) are three accidental numbers among 0 and 1. Specific characterizations are as shadows:

$$\rho_1 = 2 \times rand\,(0,1) \times a - a \tag{9}$$

$$a = |x \times sin(\frac{3\pi}{2} + sin\,(\beta \times 3\pi/2))| \tag{10}$$

$$x = x_{min} + (x_{max} - x_{min}) \times (1 - (t/T)^3)^2 \tag{11}$$

Where $x_{min} = 0.2, x_{max} = 1.2$.

Furthermore, two answers that are created at random are to be found in the following mathematical formulas. $P1_i(t)$ and $P2_i(t)$:

$$P1_i\,(t) = Lb + R4 \times (Ub - Lb) \tag{12}$$

$$P2_i\,(t) = Lb + R5 \times (Ub - Lb) \tag{13}$$

Earlier, to define parameter representation. As for R4 and R5, they are both equal to rand(1, D). The answer Pk(t) is distinct mathematically in Equation (14):

$$P_k\,(t) = L_2 \times P_p\,(t) + (1 - L_2) \times P_{rand} \tag{14}$$

$$P_{rand} = Lb + R6 \times (Ub - Lb) \tag{15}$$

where Pp, $p \in \{1, 2, \ldots, N\}$ is an arbitrary key, $R6 = rand(0, 1)$ besides L2 is a binary parameter. If $\mu_2 < 0.5$, $L_2 = 1$, the $L2 = 0$. μ_2 = rand(0, 1).

C) Learning Strategy

When it comes to improving algorithm performance, solution coverage space, and candidate individual variety, two useful strategies are opposition-based learning (OBL) and quasi-opposition learning (QOL). The OBL and QOL policies are used to update jellyfish individuals according to the likelihood phases of the algorithm[11]. This improves the solution's quality in the population, which in turn magnifies the optimization competence and further algorithm.

Both the opposition-based solutions are to be produced by using the OBL and QOL strategy to the i-th candidate separate Pi in the current population. recorded as $\widetilde{P}_i = \left(\widetilde{P}_i^1, \widetilde{P}_i^2, \ldots, \widetilde{P}_i^D\right)$ and $\breve{P}_i = \left(\breve{P}_i^1, \breve{P}_i^2, \ldots, \breve{P}_i^D\right)$ The specific appearance of the constituent is shown in Equation (16) and Equation (17)

$$\breve{P}_i^d = Lb^d + Ub^d - P_i^d \tag{16}$$

$$\breve{P}_i^d = rand\left(\frac{Lb^d + Ub^d}{2}\right), Lb^d + Ub^d - P_I^d \tag{17}$$

where P_i^d Is the site of the i-th applicant with $d-th$ dimension, Ub^d and Lb^d are limits with the d-th dimension in key space, correspondingly.

The following equation defines the refreshed Pi candidate jellyfish i-th equation, to summarise.:

$$P_i^{new} = \begin{cases} \widehat{P_i} \; if \; rand \; < p \\ \widehat{P_i} \; if \; rand \; \geq p \end{cases}, \; i = 1, 2, ...N \tag{18}$$

where p is selection likelihood, and p = 0.5.

After determining the current and novel candidates' goal values, the algorithm uses Equation (18) to construct N new jellyfish individuals. The 2N candidates are ranked according to the outcomes of the calculations, and the best N are selected to continue to the next cycle.

D) Steps of Enhanced Jellyfish Search Algorithm

Algorithm convergence besides local exploration is enhanced by the method of learning factors that use sine and cosine. The capacity to avoid falling into local optimization and to improve global exploitation is bolstered by the local escape operator method. The optimization competency is amplified by increasing the diversity of the candidate individuals through the OBL and QOL methods, which in turn enhances the clarification's quality in the populace. The EJS algorithm is an improved jellyfish search method that combines the JS algorithm with these three tactics. The EJS algorithm follows a similar pattern of stages as the JS algorithm. The most notable change is the timing of Step 5's implementation of the opposition-based learning technique.

E) Time Complexity of the EJS Algorithm

A collection of procedures for processing data and resolving programming issues is called an algorithm. It's used to the N, D, and T are the nodes where the EJS algorithm's temporal complexity is located. Throughout each iteration, the EJS algorithm follows this procedure: candidates swim with the operator is activated, candidates either actively move with sine factors or passively move within a swarm, new entities are generated using an opposition- strategy, and the best candidates are chosen to take part in the new generation's iterative process. When all of the aforementioned calculations are input, the time complexity The text above explains what T, N, and D signify.

$$\begin{aligned} O(EJS) \; = \; & O(T(O(candidate \; follow \; ocean \; current \; + \; local \; escaping \; operator) \\ & + O(passive \; motion \; + \; active \; motion) \; + \; O(Learning \; strategy))) \end{aligned} \tag{19}$$

$$O(EJS) \; = \; O(T(ND \; + \; ND \; + \; ND)) \; = \; O(TND) \tag{20}$$

2.3 Classification using Deep Learning Model

A) Dual-Number Residual Structure

There are four distinct scales in the MMDL-Net model, which is built using TensorFlow. The ResNet50 fundamental stage, which has stages with ratios of 3:4:6:3, is kept after comparing ResNet50, ResNet101, and ResNet152. As a result, class category recognition may suffer since the channel statistics of these phases are not generally suitable for closing this semantic gap. To overcome this, the number of channels in each stage is doubled, always in multiples of two, so that the hardware design may make full use of its features for better data processing efficiency and performance. By making this change, the suggested model may better respond to complicated classification problems by providing a larger feature representation space, which improves learning and capturing of the features and semantic information of the input images. This improvement boosts classification accuracy and makes the model more resistant to input image changes and noise. In each level, the channel numbers are doubled to (128, 256, 512, 1024). The network is still able to train and infer efficiently because of optimizations in contemporary hardware and the TensorFlow framework, even though adding more channels increases the model's computational cost and the number of parameters.

To be more precise, the $1 \; x \; 1$ convolutional layers to the number of filters increased to double the channels. The number of channels is reduced in each residual block using a 1×1 convolutional layer, features are extracted using a 3×3 convolutional layer, and the number of channels is increased using another 1×1 convolutional layer[12]. These three convolutional layers in a classic ResNet architecture have c1, c2, and c3 filters, correspondingly. But under the suggested model, to double the sum of channels by setting the number of filters in each of the layers to $2c1, \; 2c2, \; and \; 2c3$, correspondingly. To make each residual block in TensorFlow include more channels, the filter settings of the ResBlock class's conv1 and conv2 would be changed. The

ResNet architecture allows users to adjust the number of channel blocks according to their requirements. As an example, while establishing the proposed model, one may set filters 1 to 128 and filters 2 to 128 to raise the sum of channels in the paramount residual block from 64 to 128. This will improve ResNet's performance for classification tasks in TensorFlow by doubling the sum of channels for each residual block.

The suggested model has many residual blocks, the first of which has 64 channels[13]. The channel-doubling enhancement doubles the sum of channels in this block, taking it from 64 to 128. The channel counts of the remaining blocks are likewise doubled. Increasing the channel number from 128 to 256 in the second residual block, 512 in the third, and so on is one example. As seen in Figure 4, the suggested model makes use of two distinct kinds of residual structures.

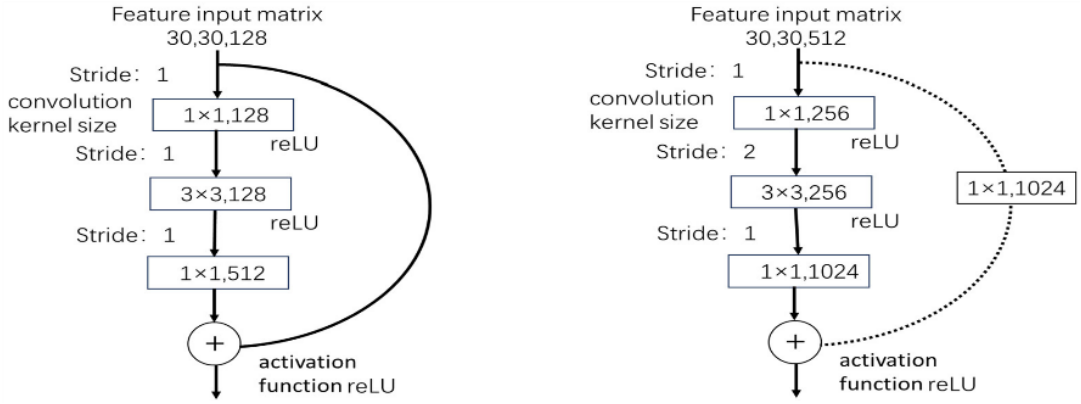

Fig 4. The two types of residual structures in the MMDL-Net model

B) Multi-Label Classification Module

To convert the sunflower leaves dataset into tensor-type image data, it is dealt with and trained within TensorFlow. The suggested model's classification process relies heavily on fully connected layers, which aggregate the feature maps produced by classification results[14]. Usually, these completely connected layers are used as input and are located at the end of the proposed design. A global average pooling operation usually follows the preliminary stages of the model, which are devoted to extraction. The shape of the feature maps is 1 x 1 $\times$ N after this process, with N being the number of channels. An N-length feature vector is the end product of flattening these feature maps. A vector representing the scores for the different classes is obtained by multiplying this flattened feature vector with a weight matrix. Afterward, instead of using the softmax function, the sigmoid function is used to activate the completely linked layer. The reason is, that in cases of multilabel categorization, a single sample could belong to more than one category at the same time. This goes against the idea behind the SoftMax function, which is best suited for jobs involving multiclass single-label classification since it presumes that each sample only be assigned to one category. When modeling a categorical probability distribution over class labels, the SoftMax function is used to make sure that for every given sample, the total of all potential labels is one. The output scores from the last layer of the network are amplified and normalized to accomplish this. By handling the existence of each label as a standalone binary classification problem, the sigmoid function offers this capability by being applied independently to each output node. In a multi-label setting, the sigmoid activation function is fundamentally useful because it handles the output from each label separately, enabling the model to depict the likelihood of each class label independently. On top of that, being a nonlinear function, it works wonders with complicated and unpredictable data.

C) Construction of the Loss Function

The loss function is utilized in TensorFlow. It is well-suited for multi-label, multi-class tasks and easily trained with gradients. Additionally, it does not require any label encoding conversion[15]. To acquire the anticipated probability for each category, the model activates the output of the layer through the sigmoid function. It uses the binary cross-entropy loss function to determine the gap between the actual labels and the anticipated probability. The cross-entropy loss function is a powerful tool for updating network parameters through backpropagation and computing the network's loss value in multi-label, multiclass tasks. For jobs with more than one class and one label, the cross-entropy loss function looks like this:

$$L = -\sum[ylogx + (1-y)\,log(1-x)] \tag{21}$$

In this case, x stands for the values that the network predicted and y for the actual labels. For multilabel multi-class jobs, the right encoding is applied to both y and x. Using the output from the network and the actual labels, the formula determines the prediction x. The cross-entropy loss formula takes the expected consequences x and labels y as inputs to get the loss value L. The backpropagation procedure is used to compute the gradient and update the network parameters based on the loss charge L.

To make sure the predicted values are in the range of 0 to 1 and that the total of the expected probability for each class is 1, the predicted values must undergo processing by an activation purpose, usually in conjunction with the function, during the calculation process [16]. For optimization, the Adam procedure is employed, which modifies the network parameters using the gradients of the parameters first- and second-order moment estimates. It optimizes network parameters efficiently and improves network performance by combining the qualities of momentum and adaptive learning rate.

3 Results and Discussion

The proposed multi-class sunflower disease detection model demonstrated strong performance across multiple evaluation metrics, achieving an accuracy of 99.56%, significantly outperforming comparable models with accuracies of 98.34% and 97.39%. This improvement can be attributed to the Enhanced Jellyfish Search Optimization Algorithm (EJFSOA), which enhanced feature selection through the integration of sine and cosine learning variables, a local escape operator, and opposition-based learning. These modifications contributed to higher precision and faster convergence in the classification process. The Multi-class Sunflower Network (MCSFNet), incorporating ResNets for feature extraction and a multi-spectral stacking module, effectively distinguished between different disease classes, as reflected by high sensitivity and F1-score values. Additional testing on noisy data confirmed the robustness of the model, with only a minor decline in performance, underscoring its resilience in practical applications. The results indicate that the proposed approach not only advances state-of-the-art accuracy in sunflower disease detection but also holds promise for real-world deployment, especially with further optimizations for low-resource environments, as discussed in the Future Work section. Figure 5 demonstrates the sample output of the planned model that correctly predicted the multiclass sunflower diseases.

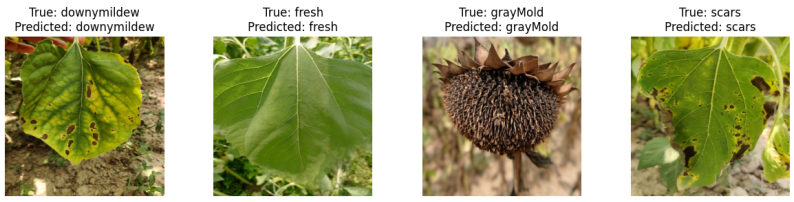

Fig 5. Sample Output of the proposed model

The anticipated model's training accuracy besides loss is mentioned in Figure 6.

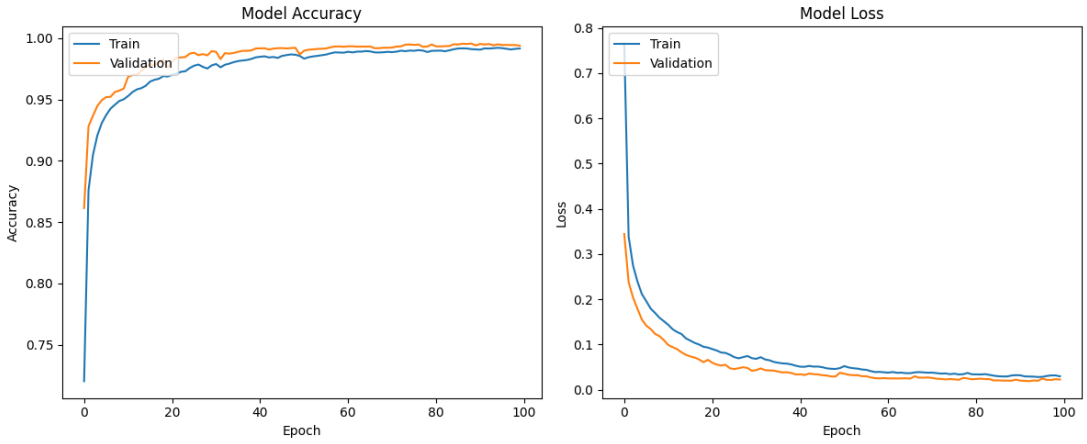

Fig 6. Accuracy and Loss of the anticipated model on training besides testing data

In the context of sunflower plant disease findings, a ROC curve is used to evaluate the performance of classification models[17]. There are the three lines typically seen on an ROC curve a Diagonal Line, a Curve Above the Diagonal, and a Curve Below the Diagonal:

Diagonal Line: This line characterizes the presentation of a random classifier, where the True Positive Rate (TPR) is equal to the False Positive Rate (FPR). It serves as a baseline, indicating no discrimination ability between classes[18].

Curve Above the Diagonal (Good Classifier): A well-performing model will have a curve that bows towards the top-left corner, representing a great FPR besides a low False Positive Rate. The area under this curve (AUC) is closer to 1, reflecting better performance.

Curve Below the Diagonal (Poor Classifier): If a curve falls below the diagonal line, it indicates a model performing worse than random guessing, with a higher False Positive Rate than True Positive Rate. This typically signals issues with the model or data.

3.1 Multi-class Performance of the Proposed Classical

Table 1. Study of proposed classical on multi-class diseases

Classifiers	Healthy	Leaf Spot	Downy Mildew	GrayMold	Scars
Accuracy	96.25	95.6	92.9	93.5	94.16
Sensitivity	88.75	98.4	86.3	87.5	85
Specificity	89.44	90.13	89.95	91.87	89.44
F1-score	90.05	92.07	92.95	93.34	91.92

In the study, the practical applicability on low-resource devices is not discussed in detail, although the complexity of the multi-layered model and EJFSOA suggests potential constraints for low-power devices. For the classifier model metrics, the accuracy for healthy leaves is 96.25, for leaf spot is 95.6, for downy mildew is 92.9, for gray mold is 93.5, and for scars is 94.16. The sensitivity metrics are as follows: healthy leaves at 88.75, leaf spot at 98.4, downy mildew at 86.3, gray mold at 87.5, and scars at 85. The specificity metrics are: healthy leaves at 89.44, leaf spot at 90.13, downy mildew at 89.95, gray mold at 91.87, and scars at 89.44. The F1-score metrics are: healthy leaves at 90.05, leaf spot at 92.07, downy mildew at 92.95, gray mold at 93.34, and scars at 91.92.

3.2 Comparative study of the Projected Feature Selection model

Table 2 provides the proportional study of the proposed FS prototypical with existing procedures in terms of diverse metrics on binary class. The Enhanced Jellyfish Search optimization algorithm (EJFSOA) without extensively comparing it to other optimization algorithms. While the benefits of EJFSOA are highlighted, the benefit from a comparative analysis with other established algorithms, such as the Whale Optimization Algorithm (WHOA), and Dwarf Mongoose Optimization Algorithm (DMOA) to strengthen claims of its efficacy. It also includes results from other popular optimization algorithms to provide a clearer benchmark. The EJFSOA's performance advantages offer readers a point of comparison of optimization algorithms.

Table 2. Comparative Analysis of Anticipated Feature Selection

Feature Selection	Accuracy	Sensitivity	Specificity	F1-score
WHOA	95.87	92.15	91.08	89.16
DMOA	96.24	93.47	93.52	90.54
JFSOA	97.26	95.16	95.03	92.63
EJFSOA	98.16	97.34	96.28	95.27

Table 2 signifies the Comparative Study of Feature Selection. The study used different feature section models as WHOA technique accuracy of 95.87 sensitivity of 92.15 and Specificity metrics of 91.08 the F1-score metrics of 89.16 correspondingly. the DMOA technique accuracy as 96.24 sensitivity as 93.47 Specificity metrics at 93.52 the F1-score metrics at 90.54 correspondingly. the JFSOA technique accuracy as 97.26 sensitivity as 95.16 Specificity metrics at 95.03 the F1-score metrics at 92.63 correspondingly. the EJFSOA technique accuracy as 98.16 sensitivity as 97.34 Specificity metrics of 96.28 the F1-score metrics of 95.27 correspondingly.

3.3 Comparative Analysis of Proposed Classifier

The analysis does not provide a detailed computational cost analysis. The EJFSOA algorithm is said to improve convergence speed, a breakdown of computational time, and resource usage it gives a better understanding of the model's efficiency, particularly for real-time applications. In the analysis of the CNN classifier technique, accuracy is 96.37 also sensitivity is 93.50, specificity of 89.18 the f1-score is 91.63 respectively. the RNN classifier technique accuracy is 97.69 also sensitivity is 95.82 and specificity of 92.62 the f1-score is 93.04 correspondingly. the LSTM classifier technique accuracy is 98.34 also sensitivity is 96.67 and specificity of 94.63 the f1-score is 95.13 correspondingly. the MCSFNet classifier technique accuracy is 99.56 also sensitivity is 98.04 specificity of 95.19 the f1-score is 96.77 respectively.

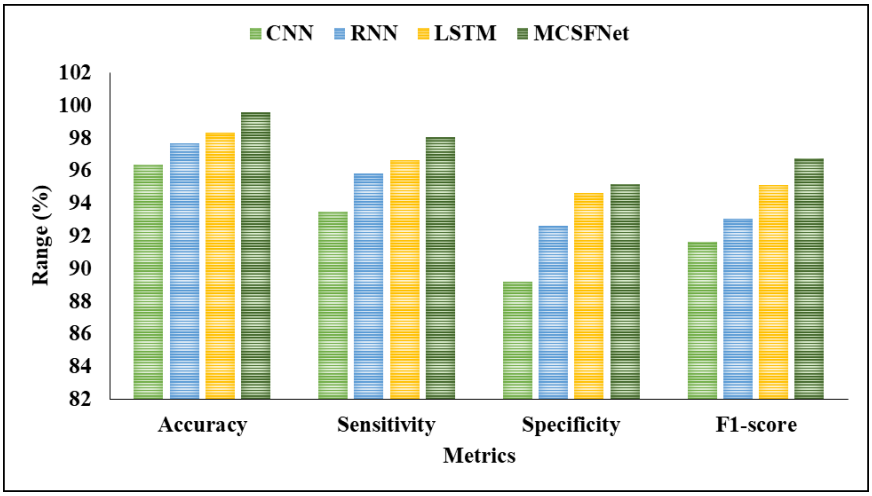

Fig 7. Graphical Description of the proposed model on the binary class of sunflower disease

4 Discussion

In this study, the proposed model for multi-class sunflower disease detection demonstrates promising results, showcasing the potential of integrating the Enhanced Jellyfish Search Optimization Algorithm (EJFSOA) with a deep learning framework like Multi-class Sunflower Network (MCSFNet). The enhancements made to the EJFSOA, particularly the incorporation of sine and cosine learning variables, the local escape operator, and opposition-based learning, significantly contributed to the improvement in feature selection and optimization. These improvements not only expedited convergence but also prevented the model from falling into local optima, which is a common challenge in complex classification tasks. The high accuracy and other performance metrics validate the efficacy of the proposed approach, positioning it as a powerful tool for disease detection. Furthermore, the robustness of the model, even with added noise, underscores its practical applicability in real-world scenarios where data quality may be compromised. However, while the model performs well on the current dataset, its applicability to larger, more diverse datasets remains a critical next step. Additionally, the model's computational cost, though manageable, may pose challenges for deployment on low-resource devices, indicating the need for further optimizations in future work.

5 Conclusion

To identify sunflower leaf illnesses that affect many classes, this study presents a deep learning model based on feature selection. To recover the accuracy of the controls and the rate of convergence for feature selection, this work suggests an improved jellyfish search (EJS) procedure. A classification network called MCSFNet is finally constructed. It integrates a residual network with TensorFlow for specific tasks, and it has two modules: one for multi-band stacking besides another for classification. It integrates a multi-label classification technique to produce by successfully concatenating data from several bands and adjusting the input and output. According to the results of the experiments, the suggested FS and classifier outperformed the previous models in some criteria, including accuracy, sensitivity, specificity, and F1-score.

Limitation and Future Scope of the Research Work

The most significant constraint of the feature selection technique is that it must be performed at a speed higher than to the quantity of period spent searching. It's possible that higher speeds to be achieved by the combination or hybridization of feature selection with other methods. To enhance the recognition rate, additional studies to involve the exploration of deep classifiers and other hybrid methods for the classification of segmented images. To enhance the generalizability and practical applicability of our multi-class sunflower disease detection model, future work will focus on validating performance across larger, more diverse datasets to ensure robust generalization under varied environmental conditions. Additional experiments with other optimization algorithms will also be conducted to gain insights into the strengths and limitations of the Enhanced Jellyfish Search (EJFSOA) algorithm relative to alternative methods. Furthermore, optimizations such as model pruning, quantization, and reduced complexity in the classification module will be implemented to lower computational costs, making the model more suitable for real-time applications on low-resource devices, including mobile and edge platforms used in agriculture. Given the impact of noisy data in real-world applications, we plan to integrate noise-reduction techniques and explore advanced augmentation strategies to enhance robustness and reliability in challenging field conditions. These future directions support our commitment to developing a reliable, adaptable, and efficient model for multi-class sunflower disease detection that can be effectively implemented in diverse agricultural contexts.

References

1) Sathi TA, Hasan MA, Alam MJ. SunNet: a deep learning approach to detect sunflower disease. In: and others, editor. 7th International Conference on Trends in Electronics and Informatics (ICOEI), 11-13 April 2023, Tirunelveli, India. IEEE. 2023;p. 1210–1216. Available from: https://ieeexplore.ieee.org/document/10125676.
2) Rani KPA, Gowrishankar S. Pathogen-based Classification of Plant Diseases: A Deep Transfer Learning Approach for Intelligent Support Systems. *IEEE Access*. 2023;11:64476–64493. Available from: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10147120.
3) Kathuria S, Azeem NA, Sharma S, Bharadwaj NK. Sunflower Disease Detection Using Ensemble Deep Learning Models. *Research Square*. 2023;p. 1–26. Available from: https://assets-eu.researchsquare.com/files/rs-2490004/v1_covered_d420910c-b0f5-44ef-9b88-93d7bf8aa1cf.pdf?c=1705558622.
4) Sohel A, Sarker MMA, Das UC, Das PK, Siddiquee SMT, Noori SRH, et al. Sunflower Disease Identification using Deep Learning: A data-driven approach. In: and others, editor. 26th International Conference on Computer and Information Technology (ICCIT), 13-15 December 2023, Cox's Bazar, Bangladesh. IEEE. 2023;p. 1–6. Available from: https://ieeexplore.ieee.org/document/10441285.
5) Yogananda GS, J AB, Ramya R, Maheswari M, Sundaram SM. Plant Leaf Disease Detection using Novel Machine Learning Approach. In: 2023 International Conference on Evolutionary Algorithms and Soft Computing Techniques (EASCT), 20-21 October 2023, Bengaluru, India. IEEE. 2023;p. 1–5. Available from: https://ieeexplore.ieee.org/document/10393299.
6) Butune AK, Şahin YS, Erdinç A, Erdoğan H. Machine Learning-Based Detection and Severity Assessment of Sunflower Powdery Mildew: A Precision Agriculture Approach. *Bursa Uludag Üniv Ziraat Fak Derg*. 2023;37(2):387–400. Available from: https://www.researchgate.net/publication/374016642_Machine_Learning-Based_Detection_and_Severity_Assessment_of_Sunflower_Powdery_Mildew_A_Precision_Agriculture_Approach.
7) Levitskaya K, Lyakh V, Afanasieva O. Identification of sunflower seed pathogens and their effect on seed germination against the background of plant damage by Septoria leaf spot. *Bulgarian Journal of Crop Science*. 2023;60(6):11–18. Available from: https://www.researchgate.net/publication/376642920_Identification_of_sunflower_seed_pathogens_and_their_effect_on_seed_germination_against_the_background_of_plant_damage_by_Septoria_leaf_spot.
8) Rana G, Singh R, Pal A, Gupta R. Enhancing Sunflower Disease Identification with CNN-SVM Integration. In: and others, editor. 3rd International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON), 29-31 December 2023, Bangalore, India. IEEE. 2023;p. 1–6. Available from: https://ieeexplore.ieee.org/document/10442779.
9) Suresh, Seetharaman K. Real-time automatic detection and classification of groundnut leaf disease using hybrid machine learning techniques. *Multimedia Tools and Applications*. 2023;82:1935–1963. Available from: https://link.springer.com/article/10.1007/s11042-022-12893-1.
10) Banerjee D, Kukreja V, Vats S, Jain V, Goyal B. AI-Driven Sunflower Disease Multiclassification: Merging Convolutional Neural Networks and Support Vector Machines. In: and others, editor. 4th International Conference on Electronics and Sustainable Communication Systems (ICESC), 06-08 July 2023, Coimbatore, India. IEEE. 2023;p. 722–726. Available from: https://ieeexplore.ieee.org/document/10193473.
11) Shahoveisi F, Gorji HT, Shahabi S, Hosseinirad S, Markell S, Vasefi F. Application of image processing and transfer learning for the detection of rust disease. *Scientific Reports*. 2023;13(1):1–12. Available from: https://www.nature.com/articles/s41598-023-31942-9.
12) Revathi V, Kavin BP, Thirumalraj A, Gangadevi E, Balusamy B, Gite S. Image-Based Feature Separation Using RBM Tech with ADBN Tech for Accurate Fruit Classification. In: and others, editor. IEEE International Conference on Computing, Power and Communication Technologies (IC2PCT), 09-10 February 2024, Greater Noida, India;vol. 5. IEEE. 2024;p. 1423–1429. Available from: https://ieeexplore.ieee.org/document/10486564.
13) Deelip MS, Govinda K. ExpSFROA-Based DRN: Exponential Sunflower Rider Optimization Algorithm-Driven Deep Residual Network for the Intrusion Detection in IOT-Based Plant Disease Monitoring. *International Journal of Semantic Computing*. 2023;17(01):5–31. Available from: https://www.worldscientific.com/doi/epdf/10.1142/S1793351X22400165.
14) Ünal Y, Dudak MN. Deep Learning Approaches for Sunflower Disease Classification: A Study of Convolutional Neural Networks with Squeeze and Excitation Attention Blocks. *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi*;13(1):247–258. Available from: https://dergipark.org.tr/en/pub/bitlisfen/issue/83698/1380995.
15) Ghosh P, Mondal AK, Chatterjee S, Masud M, Meshref H, Bairagi AK. Recognition of sunflower diseases using hybrid deep learning and its explainability with AI. *Mathematics*. 2023;11(10):1–24. Available from: https://www.mdpi.com/2227-7390/11/10/2241.
16) Zhong Y, Tong M. TeenyNet: a novel lightweight attention model for sunflower disease detection. *Measurement Science and Technology*. 2023;35(3). Available from: https://www.researchgate.net/publication/376089790_TeenyNet_A_novel_lightweight_attention_model_for_sunflower_disease_detection.

17) Chetan HR, Rajanna GS, Sreenivasa BR, Yallappa GN. Plant Disease Detection Using Deep Learning in Banana and Sunflower. *Journal of Advanced Zoology*. 2023;44(3):93–106. Available from: https://www.researchgate.net/publication/374613443_Plant_Disease_Detection_using_Deep_Learning_in_Banana_and_Sunflower.
18) Song Z, Wang P, Zhang Z, Yang S, Ning J. Recognition of sunflower growth period based on deep learning from UAV remote sensing images. *Precision Agriculture*. 2023;24:1417–1438. Available from: https://doi.org/10.1007/s11119-023-09996-6.