Plant Disease Detection using Inception V3 model and Particle Swarm Optimization

Corresponding Author

Arivuk karasan. R
Research Scholar,
Bharath Institute of Higher Education and Research
arivukkarasan@gmail.com

Dr. Raguraman. D
Associate Professor,
Bharath Institute of Higher Education and Research
raguraman.auto@bharathuniv.ac.in

Abstract

The post recognition of diseases in fruits and vegetables by farmers in India is a contributing factor in the country's declining crop yield. Farmers everywhere are suffering significant economic setbacks. The majority of the agricultural loss can be attributed to diseases that affect both plants and fruits. Farmers can boost their output by increasing their awareness of the nutritional quality of the fruits and vegetables they grow. Because of this, we are motivated to create and develop a technology that will assist farmers in detecting diseases in their crops at an early stage. The proposed research utilizes Inception V3 model which is accomplished with the help of convolutional neural networks. The performance of the Inception V3 model is enhanced by tuning its hyperparameters using Particle Swarm Optimization (PSO). The accuracy of the proposed model using Inception V3 and PSO algorithm is 0.987.

1. Introduction

Diseases that affect crops and plants generate considerable yield losses, which in turn leads to high expenses for disease management and a loss of profit for farmers all over the world. Corn is a crop that is essential on a global scale, as it makes up 95% of all feed crops in the United States [1]. It experiences yield losses of 2% to 15% due to several diseases [2]. It is essential to first precisely diagnose diseases under a variety of different settings before attempting to put into place a disease management strategy that would be effective. The more traditional methods of disease management involved laborious scouting of fields, which resulted in a delay in the identification of crop diseases. Traditional methods are not only ineffective but also subjective because they rely on the expert plant pathologists' ability to visually analyze the symptoms [3]. In order to circumvent the obstacles that are provided by conventional agricultural disease analysis, methods of disease detection that are both quick and accurate have been created through the use of computer vision-based picture analysis. On the other hand, the majority of the solutions depended on feature engineering to specify the necessary traits that corresponded to each specific condition. In order for computers to reliably identify plants, crops, and the diseases that affect them, each type of plant, crop, and disease would require its own unique set of characteristics to be derived from the plant data. As a result, there has been a recent shift toward relying more heavily on deep learning (DL) in order to build generalizable answers. Over the course of the last ten years, DL has seen a rise in popularity as a result of technological breakthroughs in memory capacity, graphical processing units (GPUs), and the availability of datasets. The use of Convolutional neural networks (CNNs) is one of the more recent methods that has been developed to combat plant diseases. CNNs can automatically acquire features from images training data which are crucial for training diseases identifying models. When defining requirements for disease management systems, the given technique enables to sidestep the requirements for feature engineering and feature extraction.

2. Related works

The detection of plant diseases typically requires DL models for relying on high quantity traits which could be derived from visible signs of the disease. Therefore, having access to high-quality image databases is really necessary. DL model-based methods have grown increasingly accepted for the diagnosis of plant diseases ever since the PlantVillage dataset was first made available [4]. In addition, newer datasets are continually made accessible to the general public. Some examples of these databases include Northern Leaf Blight (NLB) datasets [7], PlantDoc [6], and Digipathos [5].

With testing accuracies reaching up to 95.81% [8, 97.82% [9], 98.3% [10], and 99.35% [11], the PlantVillage dataset was utilized for identifying all of the 38 classes contained within the dataset, which corresponded to a total of 26 different disorders and 12 different healthy classes.

Traditional machine learning methods for instance, K Nearest Neighbors (KNN) and support vector machine (SVM) were taught by extracting features from PlantVillage by utilizing DNN. These algorithms were then used to identify different diseases with an accuracy of up to 97.82%.

Additionally, the research showed that DL-based models functioned more effectively than conventional machine learning models, which were produced by making use of manual feature extraction [9]. A testing accuracy of up to 94.3% was achieved for the identification of grape disease [12], and a testing accuracy of up to 99.18% was achieved for the identification of tomato disease [13].

Conditions matching with specific crops from PlantVillage were also found. The Digipathos dataset was yet another huge plant disease dataset which were frequently utilized in the research literature for the purpose of deep learning model training. Following the utilization of the Digipathos dataset for the purpose of training DL models [14], soybean-related disorders were detected with an accuracy of up to 93.03% during testing.

In addition, the Digipathos and PlantVillage datasets were utilized for identifying damaged lesions and leaves with testing accuracy ranging from 99.93% to 99.74% [15]. In addition, the datasets obtained from PlantVillage and Digipathos were utilized concurrently in other investigations [16], [17]. The PlantDoc dataset is a lot more manageable in terms of its size. Its little size meant that the scientific community could only make limited use of it in a variety of contexts. PlantDoc was utilized in order to correctly identify the illness [18, 19]. The majority of the datasets that have been cited in the research literature were employed in the process of locating illnesses affecting a variety of crops. On the other hand, the primary emphasis of certain investigations has been to identify individual disorders. a specialized unmanned aerial system (UAS) as well as handheld imaging equipment

The dataset of NLB-infected maize [7] was utilized for identifying NLB lesions with an accuracy of one millimeter [20]. The NLB dataset was also utilized by further investigations [21], [22], and [23] for the purpose of finding NLB disease lesions. Table 1 displays the overall outcomes that have been reported in the scientific literature.

The lack of ability of trained models for generalizing the data from a variety of diverse datasets and situations is a significant shortcoming of the currently available solutions for disease identification that are dependent on DL.

Due to a lack of variety in illumination, size, or frame, datasets like Digipathos and PlantVillage are restricted in their capacity for training robust models [24]. While significant testing accuracies exceeding 99% were observed for models that were trained utilizing the PlantVillage dataset, the testing accuracy reduced to below 50% when the images from field situations were presented to DL models that were built on the PlantVillage dataset. This was due to the fact that the PlantVillage dataset was used for training the models [25], [26]. This was despite the fact that significant testing accuracies of over 99% were recorded for models trained utilizing the PlantVillage dataset.

It was reported that testing accuracies of 99.53 percent [25] and 99.01 percent [26] were achieved while hidden testing images with respect to distinct situations were utilized from the exact dataset. After using photos obtained in the field for testing models that had been trained on images obtained in the lab, the accuracy of the tests plummeted to as low as 33.27% and 33.97% respectively. However, when models that had been trained on images taken in the field were used to test models that had been generated in the lab, an enhanced accuracy of 65.69% was seen.

The initial paper that presented the PlantDoc dataset presented the generalization of DL through training with PlantVillage and testing with the PlantDoc dataset under field conditions, that ended in testing accuracies reaching as high as 39.87%. This was accomplished by using the PlantDoc dataset for both training and testing. In addition, photos from the PlantVillage collection with consistent backdrops have been utilized as of late for DL-based disease diagnosis in agricultural settings. The models performance in terms of their ability to generalize increased from 39.38% to 72.03% [27].

3. Methodology

In this proposed methodology, the use of the Inception V3 deep learning model is investigated to detect plant diseases. The Inception V3 model is a convolutional neural network (CNN) which was pre-trained on a large dataset of images. We will use the PSO algorithm to tune the hyperparameters of the Inception V3 model to improve its performance on a dataset of plant disease images.

The following steps will be used to detect plant diseases using the Inception V3 model:

1. Pre-process the images to remove noise and other artifacts.
2. Use the PSO algorithm to tune the hyperparameters of the Inception V3 model.
3. Train the Inception V3 model on the pre-processed images.
4. Test the trained model on a separate set of images.

3.1 Convolutional Neural Network

Convolutional neural networks (CNN), are a type of multi-stage deep architecture which is alternating convolutional layers with subsampling or pooling layers, and then finishes with one or more fully connected layers. Because of its hierarchical nature, it is easier to capture layer-wise feature representations and for learning invariant features, moving up to higher layers from lower levels. (LeCun et al., 1989) illustrates in Figure 3.1, a conventional convolutional architecture that can be used for number recognition.

In this instance, the inputs are fed forward by two-phase subsampling and convolutional operations to get feature representations, and after that, a Gaussian classifier is utilized to build probability distribution. The convolutional neural network normally consists of the three primary components that are outlined further down in this paragraph.

As can be seen in Figure 3.2 (a), the convolutional operation generates the weighted sum of input values for pixels by sliding a weighted window along each pixel of the image. This is illustrated in the figure 3.2 (b). This produces the desired result of the weighted sum of input pixel values. After that, a non-linear activation operation known as the activation function is applied in order to prevent the system from learning meaningless linear representations based on the input. Rectified Linear Unit (ReLU), that is a non-negative piecewise function which consistently produces the highest value among the input and zero, is one of the most efficient activation functions. It is theoretically stated in Equation (1) as follows:

$$f(x) = \max(x, 0) = \begin{cases} x, x > 0 \\ 0, x \leq 0 \end{cases} \quad (1)$$

The input, denoted by X, is convolved with a filter, denoted by W, that has the dimensions Kx x Ky. This is the convolutional process. The output (Y) that was produced as a result can be represented quantitatively using equation (2):

$$Y = f\left(\sum_{i=1}^{n} X_i * W + b_i\right) \quad (2)$$

where *f* is the activation function, *bi* is the bias of outputs, * is the sign of convolution operation, and *n* denotes the number of elements.

In conclusion, the 2D convolutional process consists of performing element-wise multiplication among the weight matrix and the input matrix, followed by computing the weighted sum of the input pixel values.

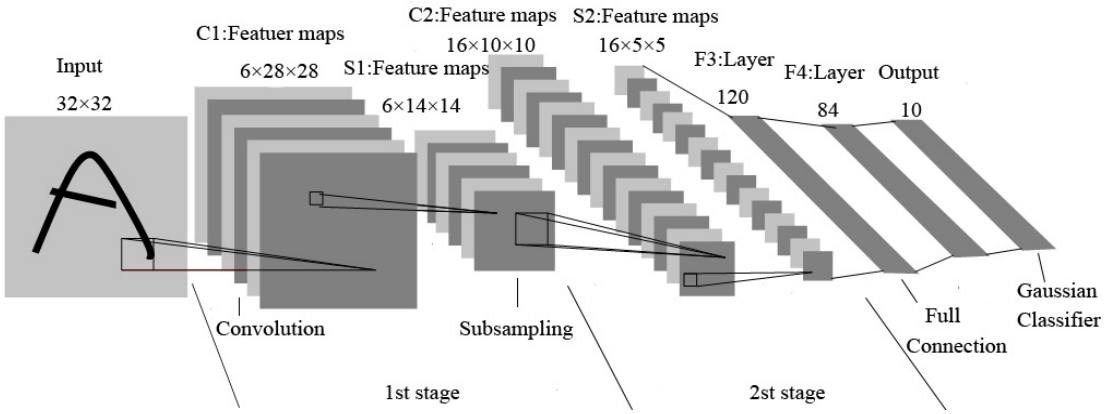


Figure 3.1 : Graphical representation of convolutional neural network

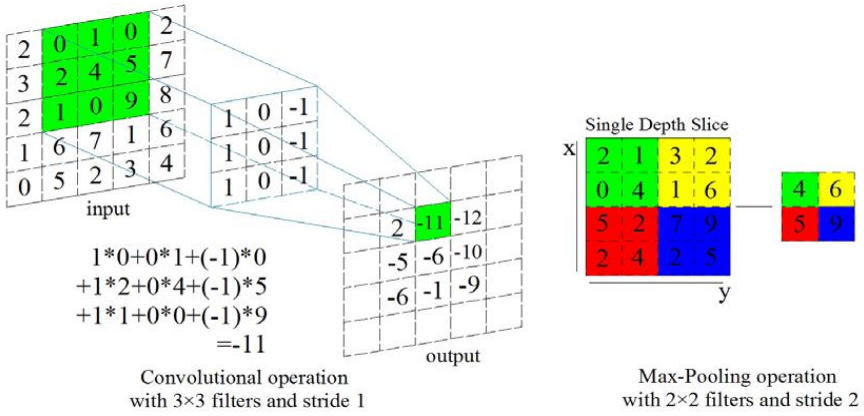


Figure 3.2 Representation of convolution and max-pooling operation: (a) convolutional operation; and (b) max-pooling operation.

The pooling process is quite similar to the convolution operation in that it subsamples features from the preceding layer and aggregates tiny pitches of pixels. This is accomplished by sliding across the pixel matrix, a weighted window. The max-pooling operation, which can be seen in Figure 3.2 (b), is one of the pooling operations that is utilized the most frequently. The max-pooling algorithm calculates the pixel value that is the highest possible over the non-overlapping portion of the weighted window. The pooling technique is frequently used for convolutional architecture since it helps for capturing significantly meaningful feature representations while also reducing the amount of processing that is required.

A number of affine transformations are typically used to convert feature mappings into 1-D feature vectors when working with fully connected layers. Following this, a classifier is implemented for providing a probabilistic distribution that is class-specific. In the field of object recognition, the SoftMax classifier is frequently used to standardize the label probability, and its mathematical description may be found in Equation (3):

$$softmax(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{n} e^{y_i}} \quad . \tag{3}$$

According to LeCun et al. (2015), a convolutional neural network possesses four distinctive keys, which are the hierarchical architecture, the shared weight, the pooling operation and the local connection. Previous research on the mechanism of the visual cortex suggests that human cognition for the real world progresses to the global level from the local. Therefore, a convolutional neural network is intended to imitate the human visual system. This means that each neuron in the network is responsible for capturing local features and integrating them with other local information in order to accurately represent the entire image in higher layers. Convolutional and max-pooling techniques, when combined with a local connection, have the potential to provide significant and one-of-a-kind feature representations of particular data. Additionally, the usage of sharing weights is made possible by the sliding weighted window implemented in pooling and convolution operations. The concept of sharing weight exemplifies that the statistical attribute for the whole image is spatial identity; to put it another way, partial features can be shared. This was demonstrated by the idea of sharing weight. As a consequence of this, it is possible for extracting features from every pixel location of the image utilizing the same weights. In addition, a hierarchical design is introduced in order to investigate correlations between neurons of adjacent layers. This is done in order to extract the layer-wise features. In contrast to a fully connected network, CNNs have a specific architecture that makes it easier to extract useful characteristics while simultaneously reducing the number of parameters and neuron connections. Finally, the convolutional neural network is the most advanced technology in the field of computer vision. It is frequently utilized for the objectives of image classification and recognition, particularly for the Inception-v3 model.

3.2 Inception V3 model

The Inception-v3 model is comprised of three distinct components: the upgraded Inception module, the basic convolutional block, and the classifier. Feature extraction makes use of the fundamental convolutional block, which consists of layers that alternate between convolutional and max-pooling.
The enhanced Inception module was developed on the basis of NetworkIn-Network (Lin et al., 2014), which is a method in which multi-scale convolutions are carried out consequently and the convolutional results of every branch are then concatenated. Auxiliary classifiers allow for more consistent training outcomes, better gradient convergence, and alleviation of simultaneously disappearing gradients and overfitting concerns. These benefits are achieved by the utilization of auxiliary classifiers.
The 1 x 1 convolutional kernel is utilized extensively throughout Inception-v3 in order to cut down on the total number of feature channels and speed up the training process. In addition, the huge convolution is broken down into a series of smaller convolutions, which brings the total cost of computing and the number of parameters down. In conclusion, the one-of-a-kind architecture of Inception contributes to Inception-v3's superior performance in the field of object identification, which places it at the forefront of the current state of the art. As a result, this model is utilized quite frequently for the purpose of transfer learning.

3.3 Particle Swarm Optimization

The exposition of the PSO algorithm that follows is considered to be relatively comprehensive. Mathematically, the PSO can be defined using the continuous space coordinate system.
The basic purpose of the PSO algorithm is to explore the search space for positions and locales which are substantially close to the global maximum or minimum solution. This is the algorithm's primary goal (s). The amount of different aspects that need to be optimised is what determines the search space size, which is also referred to as the dimension of the problem. For instance, if there are n dimensions in the search space, then there should also be in the objective function n variables to account for those dimensions. The PSO algorithm takes into account a significant number of different parameters.

While calculating a particle's fitness value at a given place, the present position of the particle is taken into account. Each particle possesses three parameters: its location ($x^i \in R^n$), its velocity, or the rate at which its position is shifting (represented by the notation v^i), and its prior best position (represented by the notation p^i). Additionally, the location of the particle that possesses the highest fitness value is referred to as the global best position, and the symbol for this location is G.
A search space point is referred to by the notation xi, which stands for the location of each individual particle. This position is referred by a set of coordinates. During the process of looking for anything, the present locations of all of the particles are assessed with the use of a fitness function. After then, of each particle the fitness value is done in comparison with the current location, and the position that yields the greatest result is saved in the list of previously best positions (p^i). Otherwise, the prior best positions are where the values of the particles are stored, and they include information about their locations.

$$x^i(t+1) = x^i(t) + v^i(t+1) \quad (4)$$

Adding the velocity to the present position of each particle causes them to travel in the following manner:

$$v^i(t+1) = wv^i(t) + c_1 r_1 (p^i(t) - x^i(t)) + c_2 r_2 (G - x^i(t)) \quad (5)$$

where w denotes the inertia weight, c1 indicates the cognition learning factor, c2 denotes the social learning factors, r1 and r2 denote the uniformly produced random numbers in the range of [0,1], and pi is the optimal solution for the i-th particle. Because the speed of the particles is determined by random variables, the movement of the particles is also completely unpredictable. The movement of the particles is therefore referred to as random walk.

3.5 Parameters used

For evaluating the performance of the proposed model using Inception V3 and PSO algorithm, the following matrics are used. For machine learning algorithms, the various performance metrices that are used to evaluate the performances are :

Precision: Precision is defined as the quantity of correctly classified data divided by the total quantity of accurately forecasted classified data in a given sample size.

$$\text{Precision} = \frac{TP}{TP+FP} \quad (6)$$

Recall: The recall statistic is evaluated as the number of correctly classified data divided by the total amount of correctly classified data.

$$\text{Recall} = \frac{TP}{TP+FN} \quad (7)$$

F1-Score: The definition of the F1-score is the harmonic mean of the precision and recall scores.

$$F1 - \text{Score} = \frac{2*(\text{Precision} * \text{Recall})}{\text{Precision} + \text{Recall}} \quad (8)$$

Accuracy: The accuracy of a dataset is evaluated by the division of the overall quantity of correct classifications on the dataset by the total number of classifications in the dataset.

$$\text{Accuracy} = \frac{TP+FN}{TP+TN+FP+FN} \quad (9)$$

4. Results and discussion

4.1 Dataset used

The dataset we will use for this project is the "New Plant Diseases Dataset" from Kaggle. This dataset is comprised of around 87 thousand RGB photos of crop leaves that are either diseased or healthy. These images have been categorized into a total of 38 different categories after being sorted. The complete dataset is then appropriately split into a training set and a testing set in the ratio of 80/20,

and throughout the process, the directory structure is maintained. In the final phase, in order to formulate a hypothesis, a new directory having an overall amount of 33 examples of test images was produced. We will use a subset of this dataset for training and a separate subset for testing. A sample of the dataset used is as given in figure 4.1.
In the proposed research 8 different types of plants are considered for prediction of diseases. The plants considered are apple, tomato, potato, cherry, chilly, corn, bell pepper and Grapes. Each plants is further classified based on their corresponding diseases.

Link: - https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset

Figure 4.1 : Sample Dataset used

The dataset has been divided in the ratio (90:10) for training and testing. 8 different plants have been taken for evaluations. The model is trained with a total of 5000 epochs of repetition. Each porch represents one round of the iterative process of forward and backward propagation. For the purpose of assessing the performance of the model, a variety of starting learning rates have been provided, and the exponential degraded rate is 0.94. In this particular investigation, a tenth of the training examples are selected at random for use as testing samples, and the model is validated after every one hundred epochs.

4.2 Preprocessing

On the dataset, proprocessing steps are performed. The accuracy of machine learning models can be improved through the use of an image pre-processing phase. When compared to a more complex model that was trained on photos that were not pre-processed, a basic model that uses pre-processed images can acquire a high level of accuracy more quickly. In the proposed research, conservative and median filtering are performed for reducing the noise.
The median filter calculates the median of the pixel intensities which is enhanced by the center pixel in a n x n kernel and then uses that median to replace the pixel intensity of the center pixel. The median filter is superior to the mean filter and the Gaussian filter when it comes to removing salt and pepper noise. However, the median filter does not address speckle noise.
The salt and pepper sounds can be removed with the help of the conservative filter. Finds the lowest possible intensity as well as the highest possible intensity within a neighborhood of a pixel. If the intensity of the center pixel is higher than the maximum value, then the center pixel's value will be changed to the maximum value. If the total value is lower than the minimum value, then the minimal value will be used in its place. Although it does not get rid of speckle noise, the conservative filter keeps the margins intact.

4.2 PSO Optimization

The PSO algorithm is used for the hyperparameter tuning of the Inception V3 model. The ability of hyperparameters to directly alter the behavior of the training algorithm is the primary reason for the importance of these parameters. The effectiveness of the model that is being trained is significantly impacted by the hyperparameters that are selected for use throughout the training process. Execution of Hyperparameter Tuning using PSO for Inception V3 Model in given in figure 4.2.

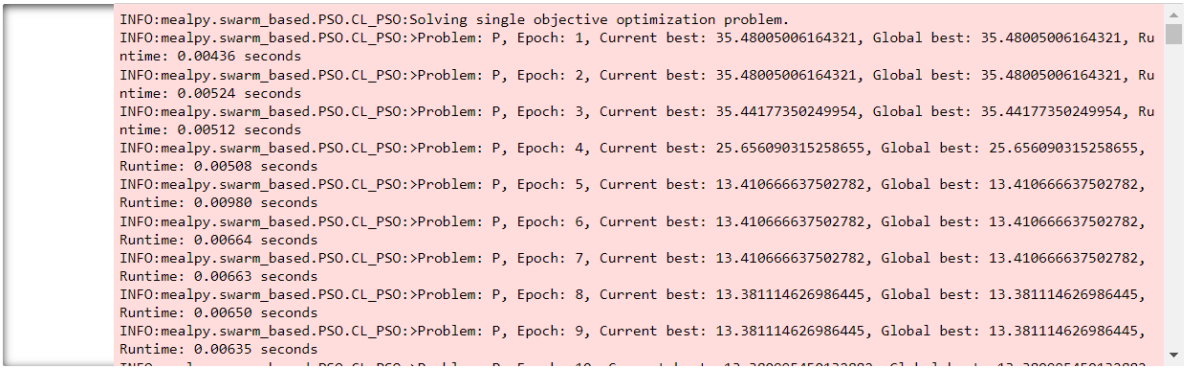


Figure 4.2 : Execution of Hyperparameter Tuning using PSO for Inception V3 Model

Best Parameters Value calculated through PSO for Inception V3 Model (Batch Size and Epoch) are
Best hyperparameters : [24.706 39.289]
Best fitnenss: 0.657

Global and Local Data Execution Trajectory per Epoch is represented in figure 4.3. This Graph is representing the trajectory per epoch regarding the time(y) taken to evaluate an optimum point(X) locally [in 1 cluster] and globally in entire data.

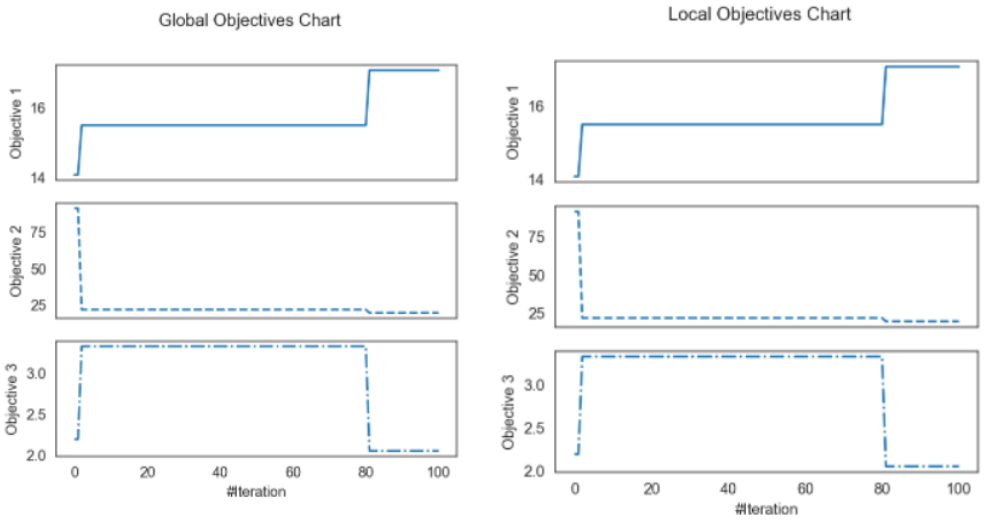


Figure 4.3 : Global and Local Objectives chart

Global and Local Best Fitness graph is represented in figure 4.4. The graphs shows the probablity of evaluating Best Optimum Points per iteration. The fitness function value is reducing to zero for both local and global best fitness as the iteration increases.

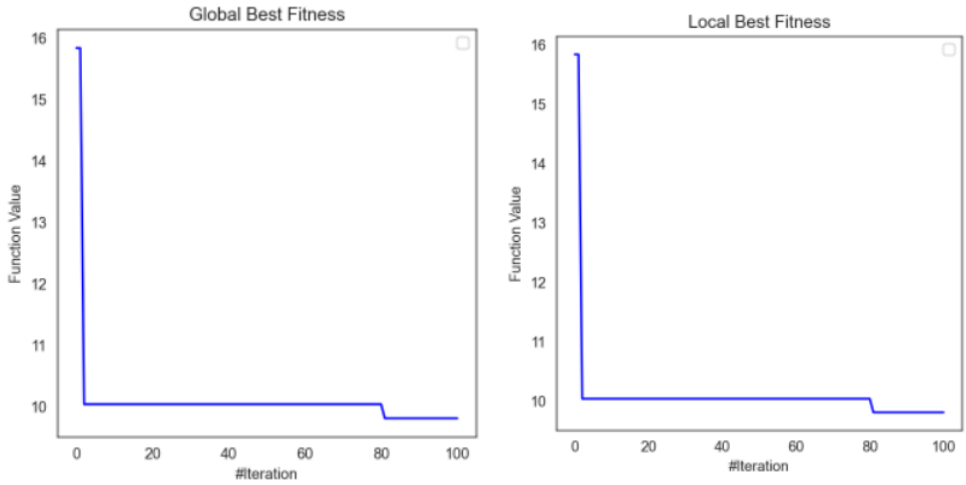


Figure 4.4 : Global and Local Best Fitness graph

Figure 4.5 shows the execution time taken for finding best optimum point per iteration, and the second graph shows the Partial Exploration for finding best optimum points and its validity.

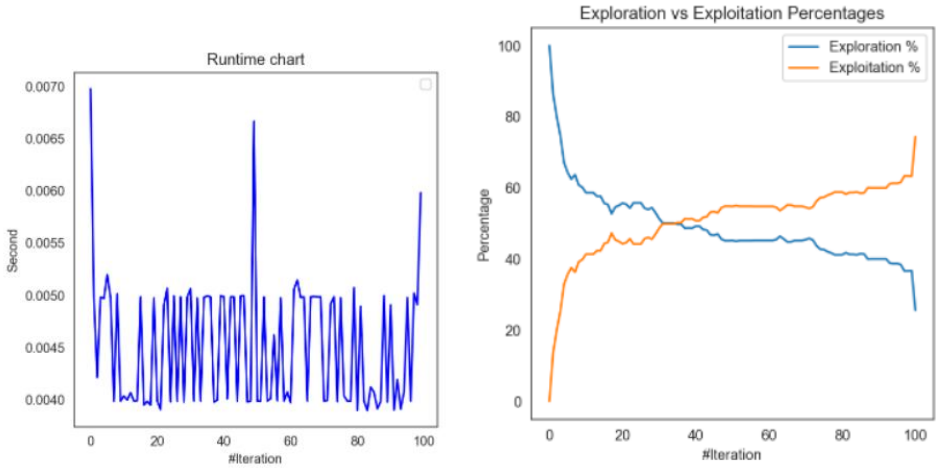


Figure 4.5 : Runtime and Exploration/Exploitation Chart

Diversity Measurement and Agent Based Trejectory Chart is represented in figure 4.6. In figure 4.6, first Graph Shows the difference of optimum points calculation ratio for each cluster per iteration, Second Graph Shows the Agent(Particles) involve for finding each data points each iteration.

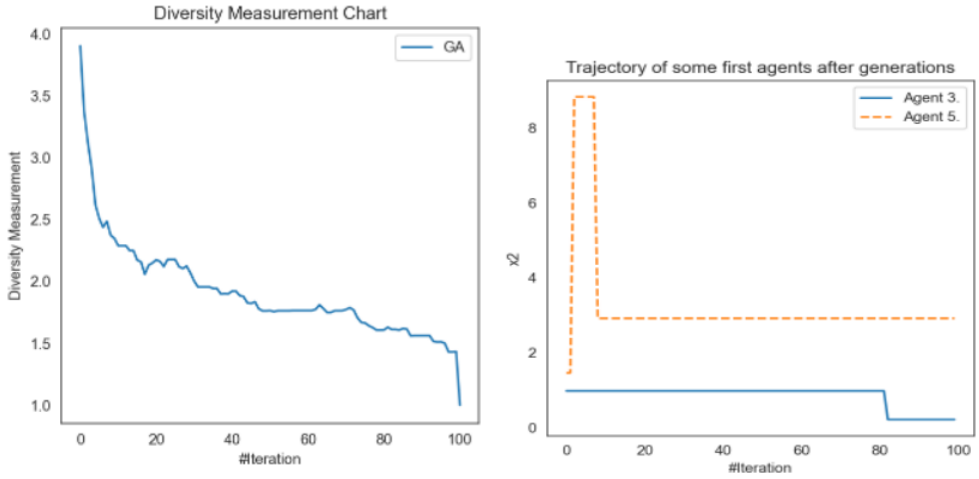


Figure 4.6 : Dencity Measurement and Agent Based Trejectory Chart

4.3 Model Training

The proposed model was trained and tested for the apple plant and the performance metrics were calculated. The model's Accuracy to Loss Ratio was evaluated and is represented graphically as represented in figure 4.7. The x-axis of the graphs are representing the number of epochs and the y axis is representing the corresponding loss or accuracy.

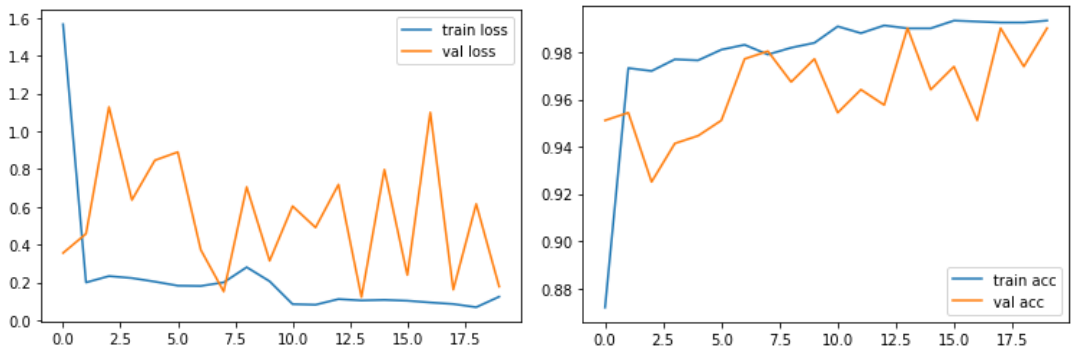


Figure 4.7 : Model's Accuracy to Loss Ratio

The model Prediction is as given

```
array([[5.2082639e-28, 0.0000000e+00, 0.0000000e+00, 1.0000000e+00],
       [0.0000000e+00, 9.4719511e-01, 0.0000000e+00, 5.2804861e-02],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 2.7980443e-12],
       ...,
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 9.2506945e-27],
       [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.0000000e+00],
       [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.0000000e+00]],
      dtype=float32)
```

```
array([1, 2, 0, ..., 1, 2, 3], dtype=int64)
```

The accuracy of the proposed model is 0.987. The PSO algorithm has enhanced the performance of Inception V3 model by tuning its hyperparameters and the model is showing very good accuracy.

Table 1 : Performance matrics of Apple plant

Performance matrics	Values
Accuracy	0.987
Precision	0.983
Recall	0.976
F1-score	0.953

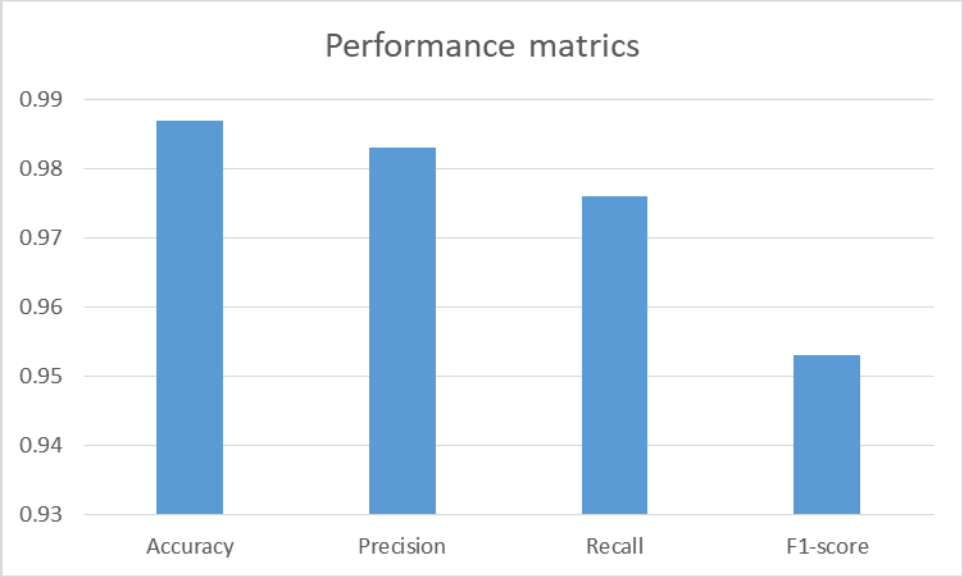


Figure 4.8 : Graphical representation of performance matrics

4.3 Accuracy to Loss graphs of other plant varieties

The proposed model was trained and tested for the tomato, potato, chilly, cherry, Bell pepper, Grapes and corn plants and the model's Accuracy to Loss Ratio were evaluated and is represented graphically as represented in figure 4.9 - 4.15. After every iteration of optimization, a model's loss value represents how poorly or how well the model behaves. An accuracy metric is utilized for quantifying the efficiency of a technique in a dependable way. The x-axis of the graphs are representing the number of epochs and the y axis is representing the corresponding loss or accuracy.

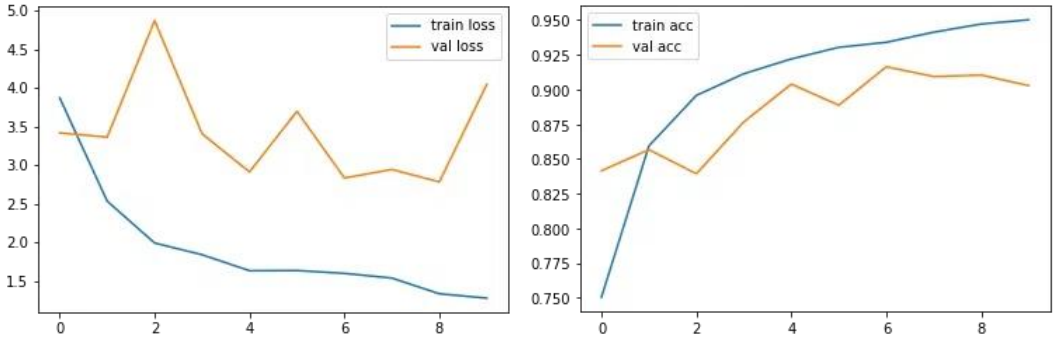


Figure 4.9 : Accuracy to Loss graph of tomato plant

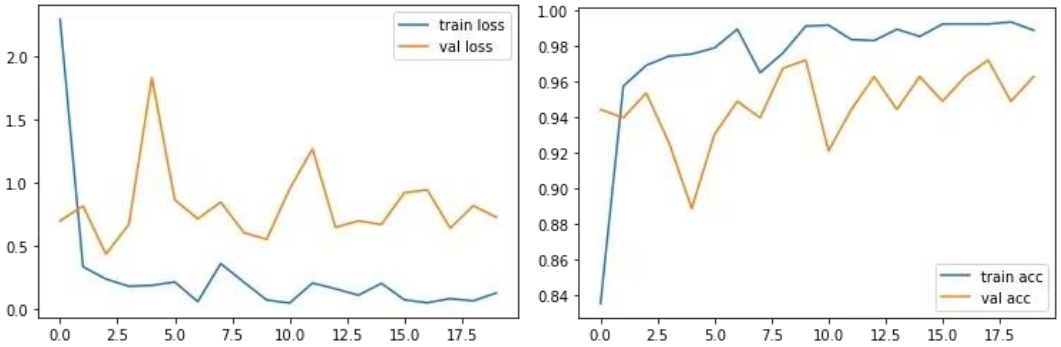


Figure 4.10 :Accuracy to Loss graph of potato plant

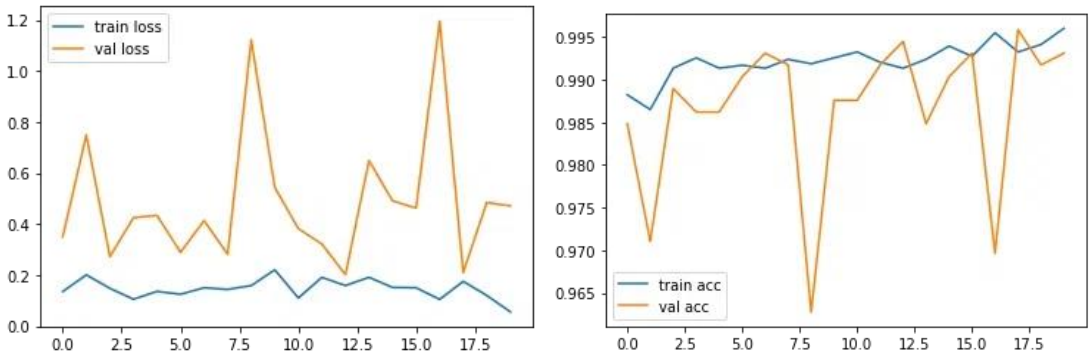


Figure 4.11 : Accuracy to Loss graph of Grapes plant

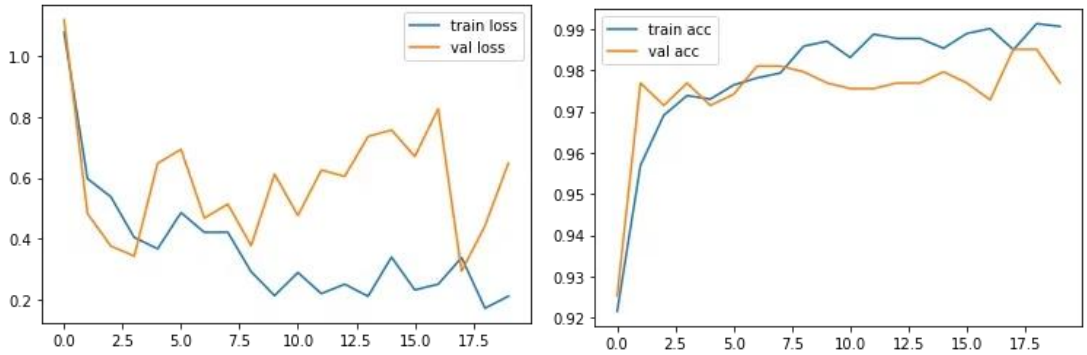


Figure 4.12 : Accuracy to Loss graph of Corn plant

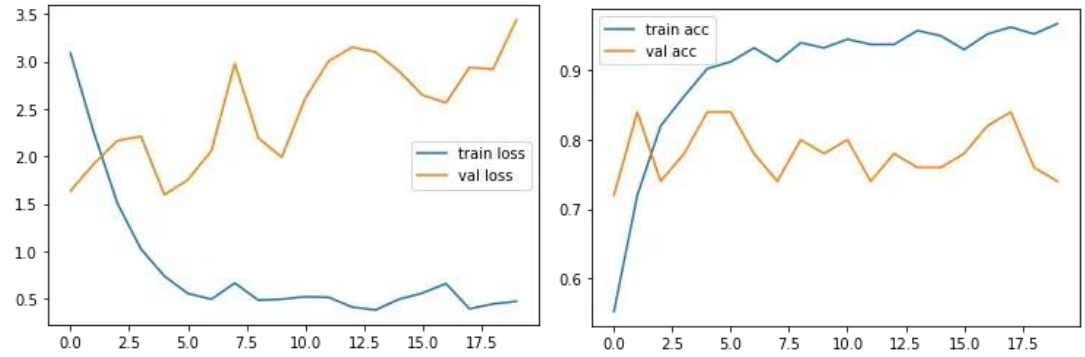


Figure 4.13 : Accuracy to Loss graph of Chilly plant

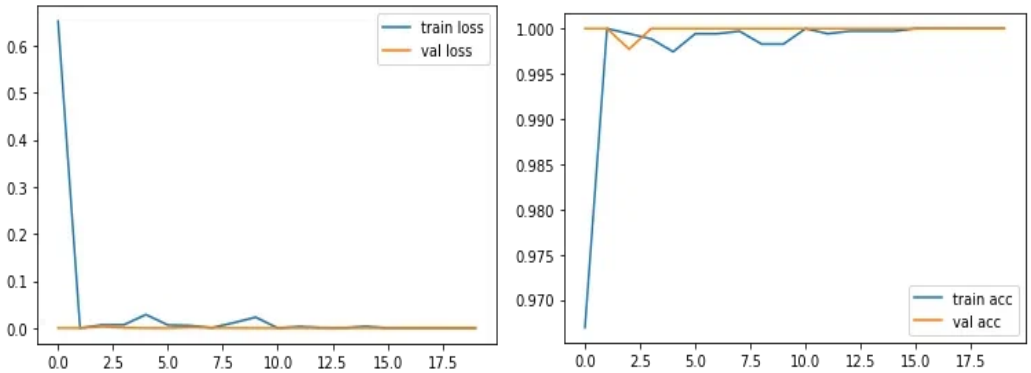


Figure 4.14 : Accuracy to Loss graph of Cherry plant

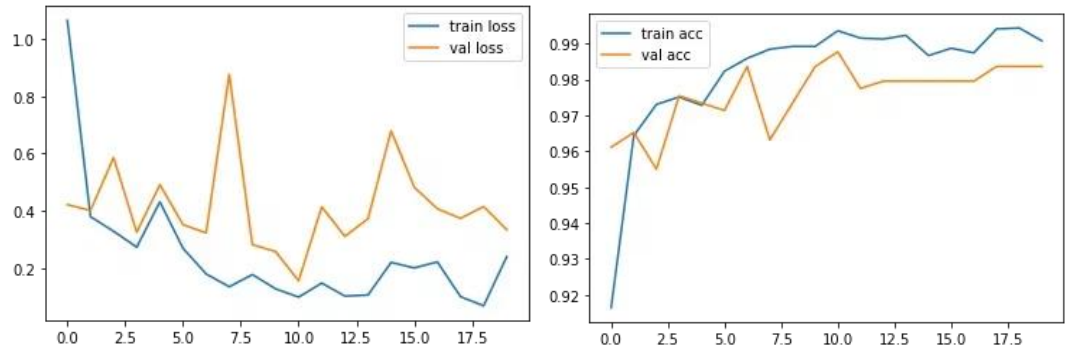


Figure 4.15 : Accuracy to Loss graph of Bell pepper plant

4.4 Individual Testing by Image

For final evaluations, an image of the diseased leaf is tested and predicted for its corresponding disease it possesses. For individual testing of the images and predicting the diseases in the corresponding plants, the images are first loaded and then converted to Numpy Array. Loading of Images and converting them into Numpy Array is as given

```
array([[[ 49.,  43.,  43.],
        [ 50.,  44.,  44.],
        [ 53.,  47.,  47.],
        ...,
        [ 70.,  64.,  64.],
        [ 67.,  61.,  61.],
        [ 64.,  58.,  58.]],

       [[ 50.,  44.,  44.],
        [ 52.,  46.,  46.],
        [ 55.,  49.,  49.],
        ...,
        [ 76.,  70.,  70.],
        [ 74.,  68.,  68.],
        [ 73.,  67.,  67.]],

       [[ 54.,  48.,  48.],
        [ 56.,  50.,  50.],
        [ 60.,  54.,  54.],
        ...,
        [ 64.,  58.,  58.],
        [ 64.,  58.,  58.],
        [ 63.,  57.,  57.]],
```

Reshaping of the Image Array takes place as (1, 224, 224, 3). Then model testing and Prediction takes place

```
1/1 [==============================] - 4s 4s/step

array([[2.6726303e-07, 4.6411334e-12, 1.0573912e-02, 9.8942578e-01]],
      dtype=float32)
```

And finally the results are predicted. In the present testing, the Apple leaf was considered. The image was uploaded and the disease was predicted as Cedar apple rust.

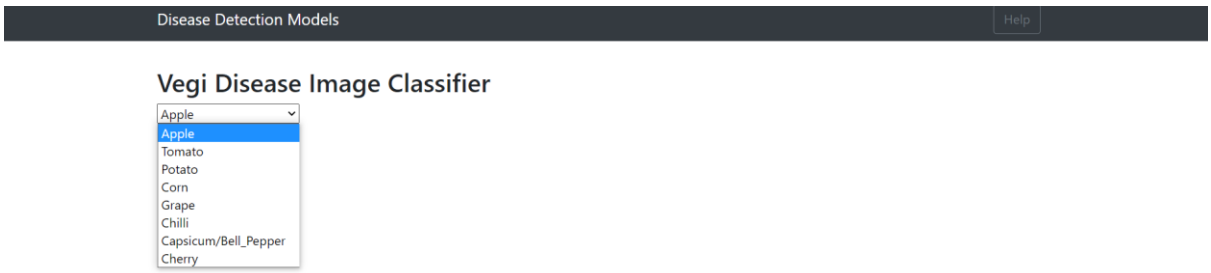
```
'Apple___Cedar_apple_rust'
```

4.4 Development of GUI Interface

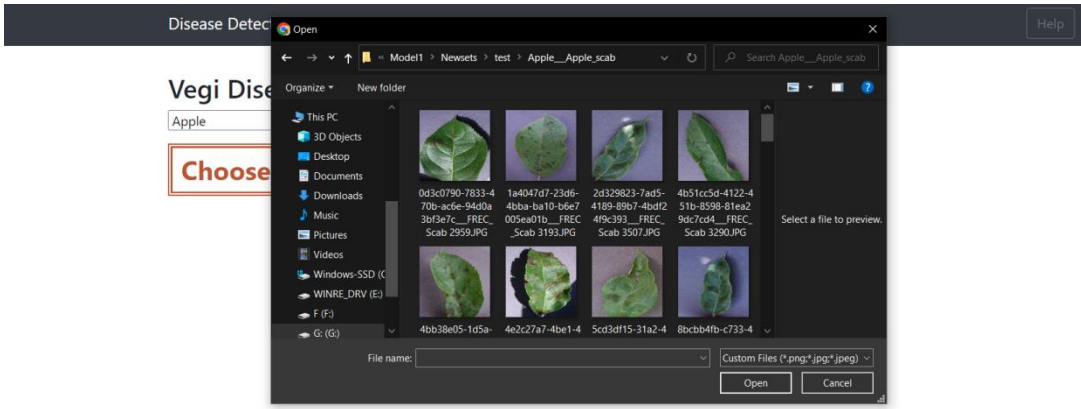
A GUI interface using Flask Python is created for checking the disease present in the corresponding leaf. The category of the leaf is first chosen in the GUI interface as given in figure 4.16 (a,b). Next the diseased leaf is uploaded. Once the diseased leaf is uploaded, the images in the dataset for the corresponding category is present in the front end model as given in figure 4.16 (c). Then the predict option is chosen as represented in figure 4.16 (d). Then the prediction takes place and the disease in the leaf is predicted as represented in figure 4.16 (e).


Disease Detection Models
Help
Vegi Disease Image Classifier
Apple
Choose...

(a)


Disease Detection Models
Help
Vegi Disease Image Classifier
Apple
Apple
Tomato
Potato
Corn
Grape
Chilli
Capsicum/Bell_Pepper
Cherry

(b)


Disease Detec
Help
Vegi Dise
Apple
Choose
Open
Model1 › Newsets › test › Apple__Apple_scab
Search Apple__Apple_scab
Organize
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows-SSD (C
WINRE_DRV (E:)
F (F:)
G: (G:)
0d3c0790-7833-470b-ac6e-94d0a3bf3e7c__FREC_Scab 2959.JPG
1a4047d7-23d6-4bba-ba10-b6e7005ea01b__FREC_Scab 3193.JPG
2d329823-7ad5-4189-89b7-4bdf24f9c393__FREC_Scab 3507.JPG
4b51cc5d-4122-451b-8598-81ea29dc7cd4__FREC_Scab 3290.JPG
Select a file to preview.
4bb38e05-1d5a-
4e2c27a7-4be1-4
5cd3df15-31a2-4
8bcbb4fb-c733-4
File name:
Custom Files (*.png;*.jpg;*.jpeg)
Open
Cancel

(c)


Disease Detection Models
Help
Vegi Disease Image Classifier
Apple
Choose...
Predict!

(e)

(f)

Figure 4.16 : Representation of the various steps in the prediction of diseases in the GUI interface.

5. Conclusion

It is recommended to forecast diseases that are present in 8 different plant kinds and then analyze how well the proposed model works. This function is quite important for the farmers, and it can also be used for a variety of other things. In the proposed research, the Inception v3 model and the Particle Swarm Optimization algorithm are utilized so that the outcomes of the classification and diagnosis of plant diseases can be improved. Some more features could be added to the model in the future, such that the combination of deep neural network and image processing is not only useful for the identification of diseases in plants, but it is also useful for the prediction of diseases in fruits and plants, and is therefore entirely useful for the agricultural sector. This is an all-encompassing benefit for the agricultural industry.

References

[1] D. S. Müeller, ‘‘Corn yield loss estimates due to diseases in the United States and Ontario, Canada from 2012 to 2015,’’ *Plant Health Prog.*, vol. 17, no. 3, pp. 211–222, 2016.
[2] G. L. Hartman, J. C. Rupe, E. J. Sikora, L. L. Domier, J. A. Davis, and K. L. Steffey, *Compendium of Soybean Diseases and Pests*. Saint Paul, MN, USA: American Phytopathological Society, 2015.
[3] C. H. Bock, G. H. Poole, P. E. Parker, and T. R. Gottwald, ‘‘Plant disease severity estimated visually, by digital photography and image analysis, and by hyperspectral imaging,’’ *Crit. Rev. Plant Sci.*, vol. 29, no. 2, pp. 59–107, Mar. 2010.
[4] D. P. Hughes and M. Salathe, ‘‘An open access repository of images on plant health to enable the development of mobile disease diagnostics,’’ 2015, *arXiv:1511.08060*.
[5] J. G. A. Barbedo, L. V. Koenigkan, B. A. Halfeld-Vieira, R. V. Costa, K. L. Nechet, C. V. Godoy, M. Lobo Jr., F. R. A. Patrício, V. Talamini, L. G. Chitarra, S. A. S. Oliveira, A. K. N. Ishida, J. M. C. Fernandes, T. T. Santos, F. R. Cavalcanti, D. Terao, and F. Angelotti, ‘‘Annotated plant pathology databases for image-based detection and recognition of diseases,’’ *IEEE Latin Amer. Trans.*, vol. 16, no. 6, pp. 1749–1757, Jun. 2018.
[6] D. Singh, N. Jain, P. Jain, P. Kayal, S. Kumawat, and N. Batra, ‘‘PlantDoc: A dataset for visual plant disease detection,’’ in *Proc. 7th ACM IKDD CoDS 25th COMAD*, Jan. 2020, pp. 249–253.
[7] T. Wiesner-Hanks, E. L. Stewart, N. Kaczmar, C. DeChant, H. Wu, R. J. Nelson, H. Lipson, and M. A. Gore, ‘‘Image set for deep learning: Field images of maize annotated with disease symptoms,’’ *BMC Res. Notes*, vol. 11, no. 1, pp. 1–3, Dec. 2018.
[8] A. S. Tulshan and N. Raul, ‘‘Plant leaf disease detection using machine learning,’’ in *Proc. 10th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2019, pp. 267–276.
[9] F. Mohameth, C. Bingcai, and K. A. Sada, ‘‘Plant disease detection with deep learning and feature extraction using plant village,’’ *J. Comput. Commun.*, vol. 8, no. 6, pp. 10–22, 2020.

[10] M. Chohan, A. Khan, R. Chohan, S. H. Katpar, and M. S. Mahar, ‘‘Plant disease detection using deep learning,’’ *Int. J. Recent Technol. Eng.*, vol. 9, no. 1, pp. 909–914, 2020.
[11] S. P. Mohanty, D. P. Hughes, and M. Salathé, ‘‘Using deep learning for image-based plant disease detection,’’ *Frontiers Plant Sci.*, vol. 7, p. 1419, Sep. 2016.
[12] P. Goncharov, G. Ososkov, A. Nechaevskiy, A. Uzhinskiy, and I. Nest-Siarenia, ‘‘Disease detection on the plant leaves by deep learning,’’ in *Proc. Int. Conf. Neuroinformatics*. Cham, Switzerland: Springer, 2018, pp. 151–159.
[13] M. Brahimi, K. Boukhalfa, and A. Moussaoui, ‘‘Deep learning for tomato diseases: Classification and symptoms visualization,’’ *Appl. Artif. Intell.*, vol. 31, no. 4, pp. 299–315, 2017.
[14] S. K. Sahu, M. Pandey, and K. Geete, ‘‘Classification of soybean leaf disease from environment effect using fine tuning transfer learning,’’ *Ann. Romanian Soc. Cell Biol.*, vol. 25, pp. 2188–2201, Mar. 2021.
[15] R. Bhagwat and Y. Dandawate, ‘‘A framework for crop disease detection using feature fusion method,’’ *Int. J. Eng. Technol. Innov.*, vol. 11, no. 3, p. 216, 2021.
[16] S. H. Lee, H. Goeau, P. Bonnet, and A. Joly, ‘‘Conditional multi-task learning for plant disease identification,’’ in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 3320–3327.
[17] S. Nandhini and K. Ashokkumar, ‘‘An automatic plant leaf disease identification using DenseNet-121 architecture with a mutation-based Henry gas solubility optimization algorithm,’’ *Neural Comput. Appl.*, vol. 34, no. 7, pp. 5513–5534, Apr. 2022.
[18] B. Anjanadevi, I. Charmila, A. Ns, and R. Anusha, ‘‘An improved deep learning model for plant disease detection,’’ *Int. J. Recent Technol. Eng. (IJRTE)*, vol. 8, no. 6, pp. 5389–5392, Mar. 2020.
[19] M. Chandra, P. S. Patil, S. Roy, and S. S. Redkar. (2020). *Classification of Various Plant Diseases Using Deep Siamese Network*. [Online]. Available: https://www.researchgate.net/profile/Manish-Chandra-3/publication/341322315_CLASSIFICATION_OF_VARIOUS_PLANT_DISEASES_USING_DEEP_SIAMESE_NETWORK/links/5ebaa82f299bf1c09ab52e48/CLASSIFICATION-OF-VARIOUS-PLANT-DISEASES-USING DEEP-SIAMESE-NETWORK.pdf
[20] T. Wiesner-Hanks, H. Wu, E. Stewart, C. DeChant, N. Kaczmar, H. Lipson, M. A. Gore, and R. J. Nelson, ‘‘Millimeter-level plant disease detection from aerial photographs via deep learning and crowdsourced data,’’ *Frontiers Plant Sci.*, vol. 10, p. 1550, Dec. 2019.
[21] C. DeChant, T. Wiesner-Hanks, S. Chen, E. L. Stewart, J. Yosinski, M. A. Gore, R. J. Nelson, and H. Lipson, ‘‘Automated identification of northern leaf blight-infected maize plants from field imagery using deep learning,’’ *Phytopathology*, vol. 107, no. 11, pp. 1426–1432, 2017.
[22] E. L. Stewart, T. Wiesner-Hanks, N. Kaczmar, C. DeChant, H. Wu, H. Lipson, R. J. Nelson, and M. A. Gore, ‘‘Quantitative phenotyping of northern leaf blight in UAV images using deep learning,’’ *Remote Sens.*, vol. 11, no. 19, p. 2209, Sep. 2019.
[23] H. Wu, T. Wiesner-Hanks, E. L. Stewart, C. DeChant, N. Kaczmar, M. A. Gore, R. J. Nelson, and H. Lipson, ‘‘Autonomous detection of plant disease symptoms directly from aerial imagery,’’ *Plant Phenome J.*, vol. 2, no. 1, pp. 1–9, Jan. 2019.
[24] M. D. S. Barros, I. D. M. Philippini, L. G. Silva, A. B. D. S. Netto, R. Blawid, E. N. D. S. Barros, and S. Blawid, ‘‘Supervised training of a simple digital assistant for a free crop clinic,’’ in *Proc. Brazilian Conf. Intell. Syst.* Cham, Switzerland: Springer, 2021, pp. 162–176.
[25] K. P. Ferentinos, ‘‘Deep learning models for plant disease detection and diagnosis,’’ *Comput. Electron. Agricult.*, vol. 145, pp. 311–318, Feb. 2018.
[26] S. H. Lee, H. Goëau, P. Bonnet, and A. Joly, ‘‘New perspectives on plant disease characterization based on deep learning,’’ *Comput. Electron. Agricult.*, vol. 170, Mar. 2020, Art. no. 105220.
[27] P. Gui, W. Dang, F. Zhu, and Q. Zhao, ‘‘Towards automatic field plant disease recognition,’’ *Comput. Electron. Agricult.*, vol. 191, Dec. 2021, Art. no. 106523.