Project Report

On

Leaf Disease Detection using MPI on Cluster and Machine Learning Classifiers

Submitted in partial fulfillment for the award of

Post Graduate Diploma in High Performance Computing

Application Programming from **C-DAC ACTS (Pune)**

Guided by:

Mr. Prakash Sinha

Presented by:

Ms. Anjali N Gaikwad	**PRN: 220940141011**
Mr. Avishkar Z Gaikwad	**PRN: 220940141008**
Mr. Prasad A Powar	**PRN: 220940141018**
Mr. Tejas M Jaronde	**PRN: 220940141029**
Mr. Krushna B Ramgude	**PRN: 220940141012**

Centre of Development of Advanced Computing (C-DAC), Pune

CERTIFICATE

TO WHOMSOEVER IT MAY CONCERN

This is to certify that

Ms. Anjali N Gaikwad

Mr. Avishkar Z Gaikwad

Mr. Prasad A Powar

Mr. Tejas M Jaronde

Mr. Krushna B Ramgude

Have successfully completed their project on

Leaf Disease Detection using MPI on Cluster and Machine Learning Classifiers

Under the Guidance of

Mr. Prakash Sinha

Project Guide **Project Supervisor**

HOD ACTS
Mr. Aditya Sinha

TABLE OF CONTENTS

1. Abstract

Farmers cultivate a lot of crops all around the year. There is a lot of effort going into this process of agriculture. But these efforts effectively get hampered when these crops are inflicted with diseases. Many factors influence disease development in plants including hybrid/variety genetics, age of the plant at the time of infection, environment, weather, genetics of the pathogen populations etc. Those crops which are affected in turn may affect other crops also which are uninfluenced at the time of initial attack. So this poses a major threat to the effort of farmers especially those who have their only way of income/livelihood based on farming. So to find an effective solution to such a major problem and curb it effectively I developed a Leaf Disease Detection system using image processing techniques and tried to improve its performance using a MPI Cluster by using cluster. In this project a performance analysis is also done to know about how much the speedup takes place when the system is run on a single machine and cluster. I have also analyzed the accuracies
Using different machine learning classifiers.

1. Introduction

Plant diseases have turned into a nightmare as it can cause significant reduction in both quality and quantity of agricultural products, thus negatively influence the countries that primarily depend on agriculture in its economy. Consequently, detection of plant diseases is an essential research topic as it may prove useful in monitoring large fields of crops and thus automatically detect the symptoms of diseases as soon as they appear on plant leaves.

Monitoring crops for to detecting diseases plays a key role in successful cultivation. The naked eye observation of experts is the main approach adopted in practice. However, this requires continuous monitoring of experts which might be prohibitively expensive in large farms. Further, in some developing countries, farmers may have to go long distances to contact experts, this makes consulting experts to very expensive and time consuming. Therefore, looking for a fast, automatic, less expensive and accurate method to detect plant disease cases is of great realistic significance.

Studies show that image processing can successfully be used as a disease detection mechanism. Since, the late 1970s, computer-based image processing technology applied in the agricultural engineering research became a common. In this study we propose and experimentally validate the significance of using machine learning techniques and MPI cluster in automatic detection of leaf diseases.

The proposed approach is image-processing-based and is composed of five main phases; in the first phase we are going to capture the image of the leaf to be tested. In the second phase, we are going to convert it into hsv image and do bitwise operation to get gray scale. Next, in the third phase, the images at hand are segmented using the MPI cluster technique using 4 threads for 4 split parts of the leaf. In the fourth phase, we calculate the texture features for the segmented infected objects. Finally, in the fifth phase, the extracted features are passed through a pre-trained machine learning model. As a testbed we use a set of leaf images taken from internet. We have tested our code on five different leaves using the proposed framework, we could successfully detect and classify the examined diseases with a precision of around 87.64% in average. The minimum precision value was 76.3%.
Present experimental results indicate that the proposed approach can significantly support accurate and automatic detection of leaf diseases.

Surveys: (10 papers)

1. Mohanty SP, Hughes DP and Salathé M (2014) Using Deep Learning for Image-Based Plant Disease Detection. Front. Plant Sci.

a) Method:

- We analyze 54,306 images of plant leaves, which have a spread of 38 class labels assigned to them. Each class label is a crop-disease pair, and we make an attempt to predict the crop-disease pair given just the image of the plant leaf.
- In all the approaches described in this paper, we resize the images to 256 × 256 pixels, and we perform both the model optimization and predictions on the downscaled images.

b) Advantages:

- Using the deep convolutional neural network architecture, we trained a model on images of plant leaves with the goal of classifying both crop species and the presence and identity of disease on images that the model had not seen before.
- Importantly, while the training of the model takes a lot of time (multiple hours on a high-performance GPU cluster computer), the classification itself is very fast, and can thus easily be implemented on a smartphone. This presents a clear path toward smartphone-assisted crop disease diagnosis on a massive global scale.

c) Limitations:

- When tested on a set of images taken under conditions different from the images used for training, the model's accuracy is reduced substantially, to just above 31%.
- The second limitation is that we are currently constrained to the classification of single leaves, facing up, on a homogeneous background.

d) Improvements:

- Image data from a smartphone may be supplemented with location and time information for additional improvements in accuracy.

2. Detection of plant leaf diseases using image segmentation and soft computing techniques (2016). Information Processing in Agriculture

a) Method:

- On image data, we have taken a color image of size m × n and every pixel have Red, Green and Blue components.
- Every chromosome shows a solution, which is a sequence of K cluster centers. Population is initialized in various rounds randomly and from existing chromosome best chromosome survives in each round for the next round processing.

b) Advantages:

- Use of estimators for automatic Initialization of cluster centers so there is no need of user input at the time of segmentation.
- The detection accuracy is enhanced with proposed algorithm.
- Proposed method is fully automatic while existing methods require user input to select the best segmentation of input image.

c) Limitations:

- The implementation still lacks in accuracy of result in some cases. More optimization is needed.
- Priori information is needed for segmentation.
- Database extension is needed in order to reach the more accuracy.

d) Improvements:

- To improve recognition rate in classification process Artificial Neural Network, Bayes classifier, Fuzzy Logic and hybrid algorithms can also be used.

3. Radha, Suja. (2017). Leaf Disease Detection using Image Processing. Journal of Chemical and Pharmaceutical Sciences

a) Method:

- **Image Acquisition:** First we need to select the plant which is affected by the disease and then collect the leaf of the plant and take a snapshot of leaf and load the leaf image into the system.
- **Segmentation:** In segmentation a digital image is partitioned into multiple segments can defined as super-pixels.
- **Low Contrast:** image pixel values are concentrated near a narrow range.
- **Contrast Enhancement:** the original image is the image given to the system and the output of the system after contrast enhancement is Enhanced Image, this is the image after removing the sharp edges.

b) Advantages:

- By computing amount of disease present in the leaf, we can use sufficient amount of pesticides to effectively control the pests in turn the crop yield will be increased.
- By using this concept, the disease identification is done for all kinds of leaves and also the user can know the affected area of leaf in percentage by identifying the disease properly the user can rectify the problem very easy and with less cost.

c) Limitations:

- Classification is done for few of the disease names in this system.
- Accuracy detection used by the algorithm was not very high.

d) Improvements:

- Accuracy can be improved by taking large dataset sample of different types of leaves on commercial scale.

4. **Dheeb Al Bashish, Malik Braik and Sulieman Bani-Ahmad (2011). Detection and Classification of Leaf Diseases using K-means-based Segmentation and Neural-networks-based Classification. Information Technology Journal, 10: 267-275**

a) Method:

- **Clustering method:** K-means clustering is used to partition the leaf image into four clusters in which one or more clusters contain the disease in case when the leaf is infected by more than one disease.
- **Feature extraction:** The method followed for extracting the feature set is called the color co-occurrence method or CCM method in short. It is a method, in which both the color and texture of an image are taken into account, to arrive at unique features, which represent that image.
- **Co-occurrence methodology for texture analysis:** The image analysis technique selected for this study was the CCM method. The use of color image features in the visible light spectrum provides additional image characteristic features over the traditional gray-scale representation.

b) Advantages:

- Identify disease type in addition to disease detection
- Deal with more diseases Be directly expanded to cover even more diseases
- Detect diseases that infect plant leaves and stems. Their proposal can identify and classify diseases that infect the stem part of plants as well.

c) Limitations:

- In an outdoor application, elimination of intensity altogether may have an effect on the classification, since the ambient variability in outdoor lighting is not taken into consideration.

d) Improvements:

- For future research, they have been some directions, such as, developing better segmentation technique; selecting better feature extraction and culling classification algorithms.

5. Guiling Sun, Xinglong Jia, and Tianyu Geng. Plant Diseases Recognition Based on Image Processing Technology (2018). Journal of Electrical and Computer Engineering

a) Method:

- **Iterative Method:** The Iterative Method can calculate the threshold in a certain extent automatically. For the iterative process, the Iterative Method includes a prior knowledge concerning the image and noise statistics.
- **Improved Histogram Segmentation Method (Calculate Threshold Automatically):** Traditional 2-Mode Method needs to set the threshold manually. As the user has huge task burden, it lacks identification efficiency. This proposed segmentation method can automatically determine the threshold.

b) Advantages:

- It can greatly reduce the user's task burden and optimize the image segmentation process.
- It is fast, efficient, and accurate.

c) Limitations:

- Has problems while handling gray statistic, especially when the ratio of the target lesion area to the background area is very small

d) Improvements:

- Increase the training images, for more accurate results, which proves the accuracy and good potential of this system.
- Can be modified easily by changing the independent and dependent variables for greater potential and improvement

6. Manu BN (2020). Plant Leaf Disease Detection and Classification using Multiclass SVM Classifier, MATLAB Central File Exchange.

a) Method:

- **Otsu Threshold Algorithm:** Thresholding creates binary images from grey-level images by setting all pixels below some threshold to zero and all pixels above that threshold to one. The Otsu algorithm is as follows:
  - i) According to the threshold, separate pixels into two clusters
  - ii) Then find the mean of each cluster.
  - iii) Square the difference between the means.
  - iv) Multiply the number of pixels in one cluster times the number in the other.

b) Advantages:

- The infected leaf shows the symptoms of the disease by changing the color of the leaf. Hence the greenness of the leaves can be used for the detection of the infected portion of the leaf.

c) Limitations:

- The implementation still lacks in accuracy of result in some cases. More optimization is needed.

d) Improvements:

- The use of ANN methods for classification of disease in plants such as self-organizing feature map, back propagation algorithm, SVMs etc. can be used to improve accuracy efficiently.

7. **Saradhambal.G, Dhivya.R, Latha.S, R. Rajesh (2018). Plant Disease Detection and Its Solution Using Image Classification. International Journal of Pure and Applied Mathematics**

a) Method:

- Feature extraction is the important part to gracefully predict the infected region. Here shape and textural feature extraction is done.
- The shape-oriented feature extraction like Area, Color axis length, eccentricity, solidity and perimeter are calculated.
- Similarly, the texture-oriented feature extraction like contrast, correlation, energy, homogeneity and mean. Leaf image is captured and processed to determine the health of each plant.

b) Advantages:

- This project implements an innovative idea to identify the affected crops and provide remedy measures to the agricultural industry.

c) Limitations:

- Complexity to convert RGB image to HSV image is very high and takes time.

d) Improvements:

- As future enhancement of the project is to develop the open multimedia (Audio/Video) about the diseases and their solution automatically once the disease is detected.

8. **Sushil R. Kamlapurkar (2015). Detection of Plant Leaf Disease Using Image Processing Approach. International Journal of Scientific and Research Publications**

a) Method:

- Feature Extraction: Features are extracted from image using **Gabor filtering** method.
- The features correspond to color characteristics are the mean and variance of the gray level of the red, green and blue channel of the spots; and other features correspond to morphological and geometrical characteristics of the spots.
- For classification purpose **Artificial Neural Network** is used.

b) Advantages:

- The proposed system is capable of detecting the disease at the earlier stage as soon as it occurs on the leaf.
- It can provide the help for a person having less knowledge about the disease.

c) Limitations:

- The proposed model cannot process large dataset of leaves into consideration.

d) Improvements:

- More images of leaves have to be used in the project to obtain more accurate results.

9. **Sladojevic S, Arsenovic M, Anderla A, Culibrk D, Stefanovic D (2016). Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification. Comput Intell Neurosci.**

a) Method:

- Training the deep **convolutional neural network** for making an image classification model from a dataset described was proposed.
- There are several well-known state-of-the-art deep learning frameworks, such as Python library Theano and machine learning library that extends Lua, Torch7.
- In addition, there is Caffe, an open source deep learning framework developed by the BVLC containing reference pretrained **CaffeNet model**. For the purpose of this research, this framework was used, along with the set of weights learned on a very large dataset.
- **CaffeNet** is a deep CNN which has multiple layers that progressively compute features from input images. Specifically, the network contains eight learning layers and five convolutional and three fully connected layers.

b) Advantages:

- This application will serve as an aid to farmers (regardless of the level of experience), enabling fast and efficient recognition of plant diseases and facilitating the decision-making process when it comes to the use of chemical pesticides.
- Achieve a valuable impact on sustainable development, affecting crop quality for future generations.

c) Limitations:

- Fine-tuning has not shown significant changes in the overall accuracy.

d) Improvements:

- The main goal for the future work will be developing a complete system consisting of server-side components containing a trained model and an application for smart mobile devices with features such as displaying recognized diseases in fruits, vegetables, and other plants, based on leaf images captured by the mobile phone camera.
- Furthermore, future work will involve spreading the usage of the model by training it for plant disease recognition on wider land areas, combining aerial photos of orchards and vineyards captured by drones and convolution neural networks for object detection.

10.Kumar KV and Jayasankar T. (2019) An identification of crop disease using image segmentation. Int J Pharm Sci & Res

a) Method:

- **Crop Disease Identification Using Deformable Method**: Deformable models are normally semi-automatic. However, two issues can appreciably affect their performance and therefore, need to be well defined: the initial conditions and the values of the parameters used. The RGB color space is converted so that the color information contained in the images can be used effectively to differentiate normal crop image and affected images. The differences in the color channels are combined to define the speed function and the stopping criterion of the deformable model.
- The methodology of the proposed system can be broken down into two segments:
- (a)**Image Processing Segment:** Where the properties of the leaf image will be enhanced segmented from the background and
- (b)**Pattern Recognition Segment:** Where the required features will be extracted and this information will be matched with the predefined knowledge about the plant diseases for detecting which disease has actually affected the plant.

b) Advantages:

- Highest segmentation accuracy
- High sensitivity and specificity
- More robust
- High computation efficiency.

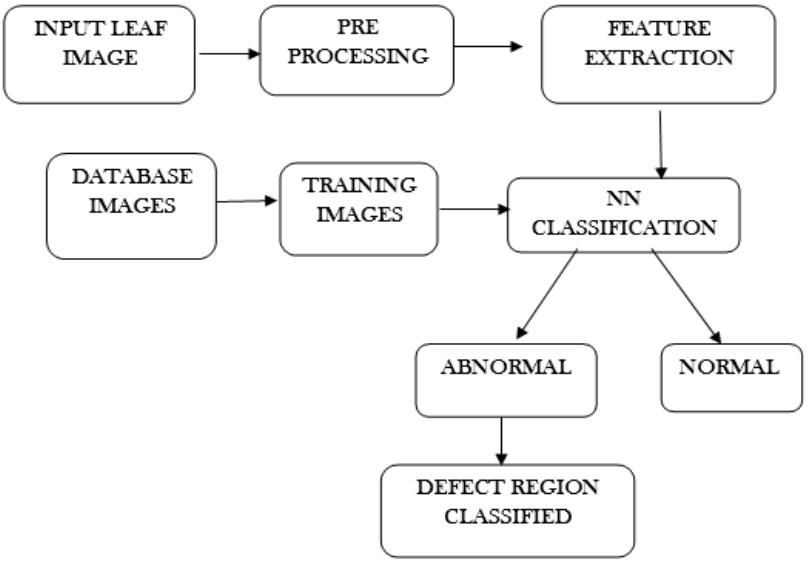
c) Limitations:

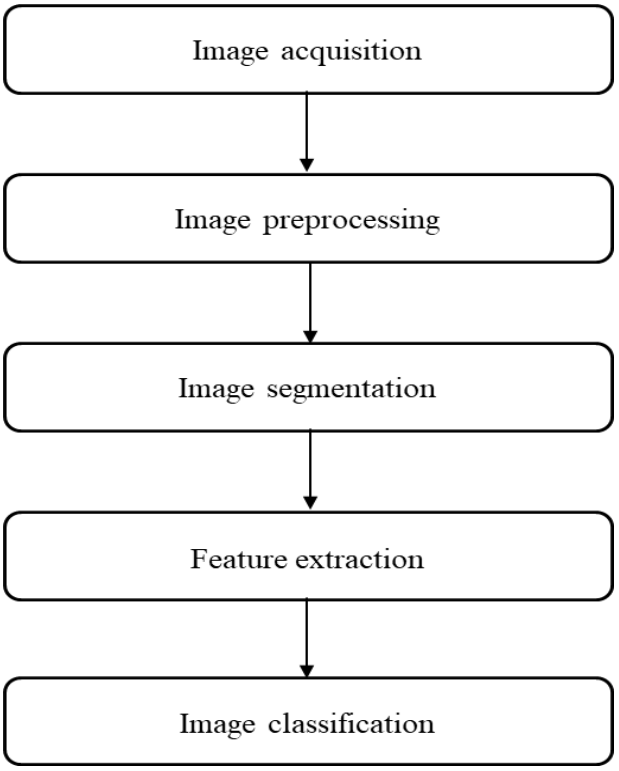
- Hugh amount of processing power is required to extract unhealthy part of leaf.

d) Improvements:

- In the near future, this suggested system will be capable of showing physical results.
- In future we have there is an idea to identify the specific diseases of the crop and also display the mechanism and chemicals to resolve it. It is better if it implemented in hardware.

Proposed Model:





❖ Training the ML model for leaf disease

1. Image Segmentation

First, we take a set of 10 training images. With the images we have we are trying to segment the part which has disease present in it. For this purpose, we first convert the image to HSV format. There we take a range of HSV values (which was obtained after histogram analysis) b/w which are generally the pixel values for diseased part. Then we do bitwise and with the original image to get RGB version of diseased part

2. Feature Extraction

We then convert the image obtained from step 1 into Grayscale image. From that image we obtain the gray level co-occurrence matrix from wherein we get features like contrast, energy etc.
Our project uses the following features:
Mean, Standard deviation, Variance, Energy, Contrast, Smoothness, Homogeneity, Entropy and RMS value

3. Training the Machine Learning Model

We already have the set of features for all 10 images obtained now. Along with that the data of whether the leaf is healthy / unhealthy was obtained already. Using the features and the result obtained we train a machine learning model which helps us to figure out for a given image of leaf whether the leaf is healthy/unhealthy using Logistic Regression model. We used it because it gave very high results while execution for both training and test data.

❖ Parallelizing the application of ML model for leaf disease detection for the image using MPI

1. Splitting of test image and assigning to slave processes

We split the test image of leaf on which we are going to apply the model to obtain the required results into 4. We obtain the center point of the image from which we get the required images. We then assign these to individual slave processes each of the images.

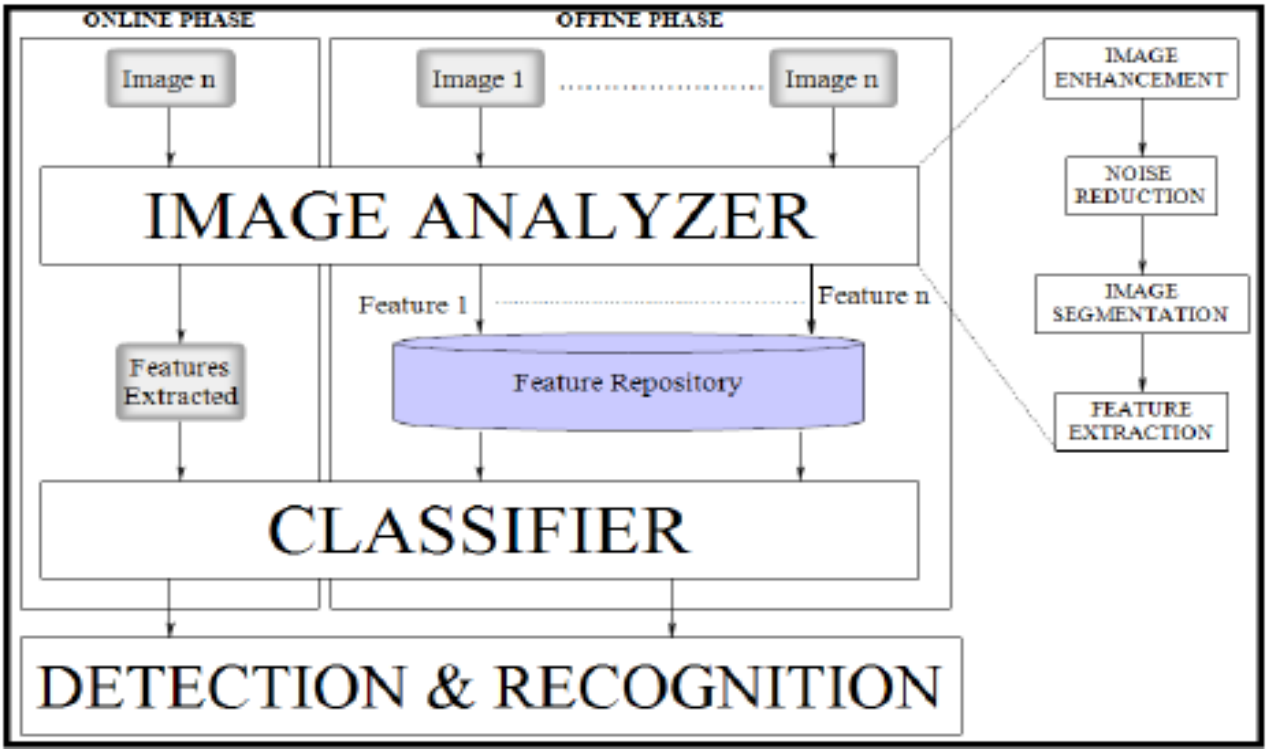
2. Application of ML model on each thread's image after feature extraction

We extract features which we discussed earlier in the previous sections from each process's assigned image based on the same procedure. Then we apply the ML model to get the result status of that image which is assigned to that particular image. We then send the result status to the main master process.

3. Reduction to obtain the final result

Now based on the image status obtained from slave processes we assign whether the total leaf is unhealthy / healthy. The reduction process is based on the fact that even if one part is unhealthy the whole leaf should be detected unhealthy.

❖ Architectural Diagram:



Steps:

1. **'n'** number of leaf images are taken from online dataset
2. The images are analyzed using a **custom image analyzer** (image processing classifier in my project)
3. After analysis, features from the images are extracted
4. From 'n' images only the **unique features** are kept, the rest are discarded
5. Features are stored in the **feature repository**
6. The **image processing classifier** classifies the images based on the different features extracted and **removes the noise** to **increase accuracy** of detection
7. Once the images are classified, we can identify which part of the leaf is infected and in turn which whole leaf has been infected.

Procedure

Step 1. MPI Cluster

- We need the install stack directory to avail the package in module
- We need to load the modules and packages in cluster
  Like ml spack
- source /opt/ohpc/pub/apps/spack/share/spack/setup-env.sh
- this is a source of creating a environment
- ml load oneapi/mpi/2021.3.0

This module used to a set environment of MPI and OPENAPI

Step 2.

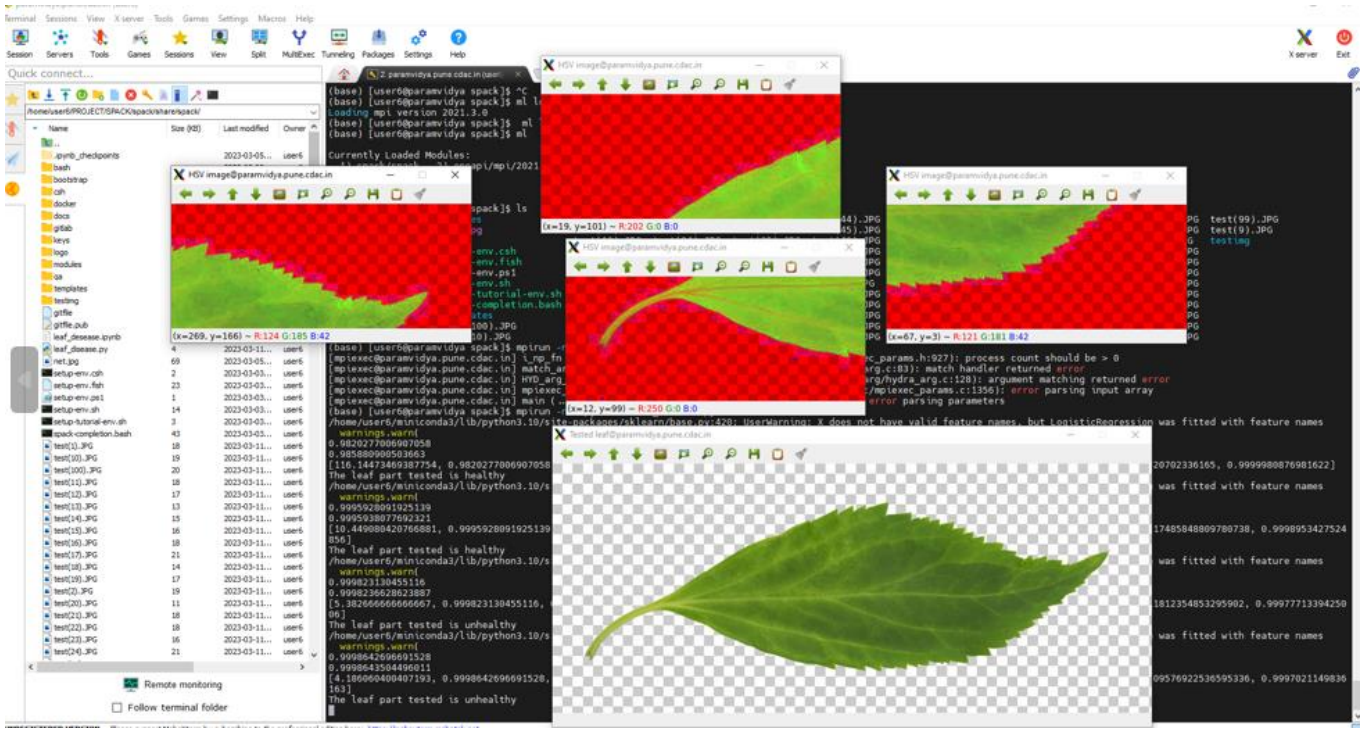
This has five phases:

1. First phase: we are going to capture the image of the leaf to be tested.
2. Second phase: we are going to convert it into hsv image and do bitwise operation to get gray scale.
3. Third phase: the images at hand are segmented using the MPI cluster technique using 4 threads for 4 split parts of the leaf.
4. Fourth phase: we calculate the texture features for the segmented infected objects.
5. Fifth phase: the extracted features are passed through a pre-trained machine learning model.

1. Output

- The number of processes used are 5
- The code will split the original leaf image into 4 parts
- There will be 4 processes to process for each leaf image part and once all the parts are tested the results will be returned to the master thread and displayed whether the leaf is healthy or not

```
(base) [user6@paramvidya spack]$ mpirun -n 5 python leaf_disease.py
/home/user6/miniconda3/lib/python3.10/site-packages/sklearn/base.py:420: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
0.9820277006907058
0.985880900503663
[116.14473469387754, 0.9820277006907058, 0.985880900503663, 2.3659269313426083, 19.458333431776175, 378.6267399421784, 0.1575718047250128, 1.347720702336165, 0.9999980876981622]
The leaf part tested is healthy
Time taken:  2.9196219444274902
/home/user6/miniconda3/lib/python3.10/site-packages/sklearn/base.py:420: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
0.999823130455116
0.9998236628623887
[5.382666666666667, 0.999823130455116, 0.9998236628623887, 0.020296346567130415, 1.960866953900896, 3.8449992109005784, 0.0014322852823029 11, 0.11812354853295902, 0.9997771339425006]
The leaf part tested is unhealthy
Time taken:  12.064669847488403
/home/user6/miniconda3/lib/python3.10/site-packages/sklearn/base.py:420: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
0.9998642696691528
0.9998643504496011
[4.186060400407193, 0.9998642696691528, 0.9998643504496011, 0.015147822162040171, 1.6910202170427797, 2.85954937444741, 0.0010749909220781595, 0.09576922536595336, 0.9997021149836163]
The leaf part tested is unhealthy
Time taken:  14.176017999649048
/home/user6/miniconda3/lib/python3.10/site-packages/sklearn/base.py:420: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
0.9995928091925139
0.9995938077692321
[10.449080420766881, 0.9995928091925139, 0.9995938077692321, 0.043123448431505305, 2.788811186469741, 7.777467833778764, 0.0033488428728269 55, 0.17485848809780738, 0.9998953427524856]
The leaf part tested is healthy
Time taken:  28.61819577217102
```



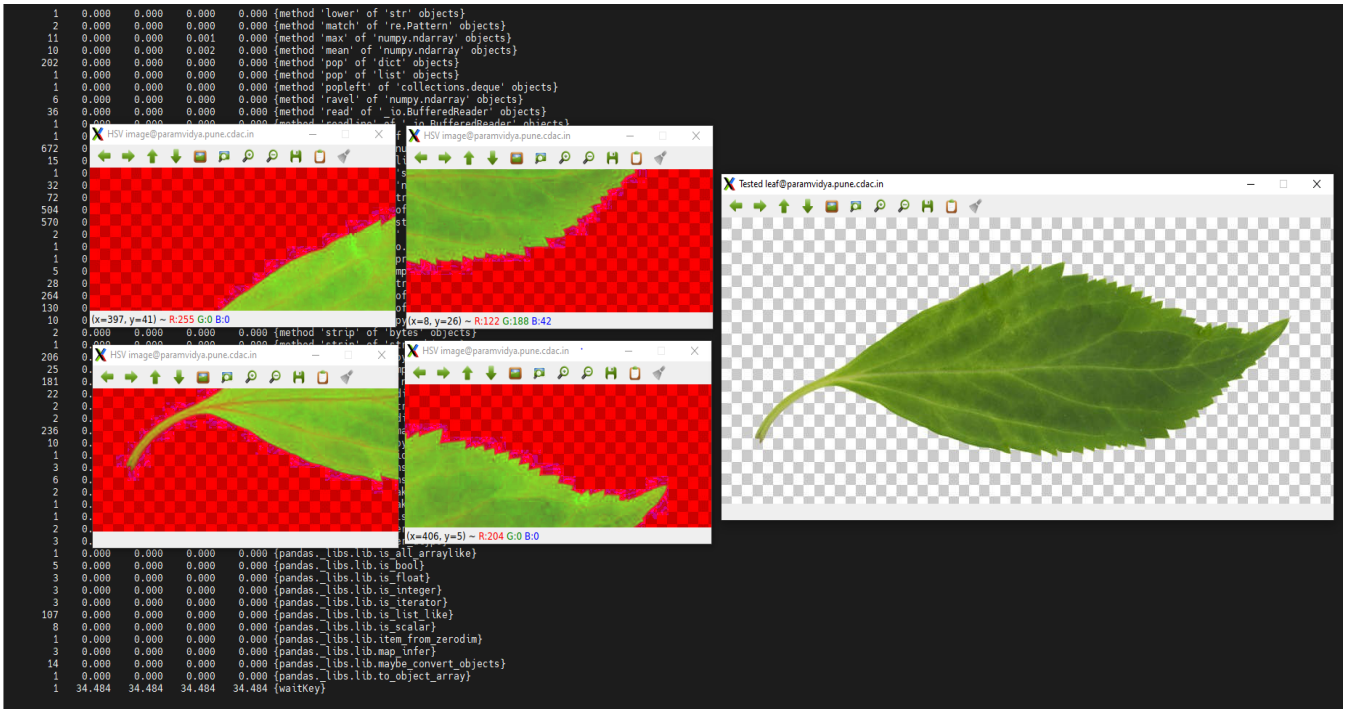
Results and Performance Analysis:

- After profiling result we are showing 3 image performance as shown in figure below:

```
(base) [user6@paramvidya spack]$ mpirun -n 5 python leaf_disease.py
/home/user6/miniconda3/lib/python3.10/site-packages/sklearn/base.py:420: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
0.9995928091925139
0.9995938077692321
[10.449080420766881, 0.9995928091925139, 0.9995938077692321, 0.043123448431505305, 2.788811186469741, 7.777467833778764, 0.0033488428728269 55, 0.17485848809780738, 0.9998953427524856]
The leaf part tested is healthy
         58304 function calls (57747 primitive calls) in 2.666 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
      196    0.000    0.000    0.001    0.000 <__array_function__ internals>:177(all)
        5    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(any)
       98    0.000    0.000    0.001    0.000 <__array_function__ internals>:177(array_equal)
      197    0.000    0.000    0.001    0.000 <__array_function__ internals>:177(atleast_1d)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(atleast_2d)
       11    0.000    0.000    0.008    0.001 <__array_function__ internals>:177(bincount)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(can_cast)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(clip)
      197    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(copy)
       21    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(copyto)
       12    0.000    0.000    0.018    0.001 <__array_function__ internals>:177(cumsum)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(delete)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(einsum)
      294    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(empty_like)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(expand_dims)
        2    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(in1d)
        2    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(linspace)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(may_share_memory)
       10    0.000    0.000    0.002    0.000 <__array_function__ internals>:177(mean)
        8    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(ndim)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(ptp)
        5    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(reshape)
        5    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(result_type)
        2    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(searchsorted)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(setdiff1d)
       88    0.000    0.000    0.004    0.000 <__array_function__ internals>:177(sum)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(take)
       17    0.000    0.000    0.062    0.004 <__array_function__ internals>:177(unique)
       55    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:100(acquire)
    50/3    0.000    0.000    1.814    0.605 <frozen importlib._bootstrap>:1022(_find_and_load)
 239/237    0.000    0.000    1.779    0.008 <frozen importlib._bootstrap>:1053(_handle_fromlist)
       55    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:125(release)
       50    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:165(__init__)
       50    0.000    0.000    0.001    0.000 <frozen importlib._bootstrap>:169(__enter__)
       50    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:173(__exit__)
       55    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:179(_get_module_lock)
```

```
/home/user6/miniconda3/lib/python3.10/site-packages/sklearn/base.py:420: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
0.999823130455116
0.9998236628623887
[5.382666666666667, 0.999823130455116, 0.9998236628623887, 0.020296346567130415, 1.960866953900896, 3.8449992109005784, 0.0014322852823029 11, 0.11812354853295902, 0.9997771339425006]
The leaf part tested is unhealthy
         58304 function calls (57747 primitive calls) in 8.285 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
      196    0.000    0.000    0.001    0.000 <__array_function__ internals>:177(all)
        5    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(any)
       98    0.000    0.000    0.001    0.000 <__array_function__ internals>:177(array_equal)
      197    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(atleast_1d)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(atleast_2d)
       11    0.000    0.000    0.006    0.001 <__array_function__ internals>:177(bincount)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(can_cast)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(clip)
      197    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(copy)
       21    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(copyto)
       12    0.000    0.000    0.014    0.001 <__array_function__ internals>:177(cumsum)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(delete)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(einsum)
      294    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(empty_like)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(expand_dims)
        2    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(in1d)
        2    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(linspace)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(may_share_memory)
       10    0.000    0.000    0.001    0.000 <__array_function__ internals>:177(mean)
        8    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(ndim)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(ptp)
        5    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(reshape)
        5    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(result_type)
        2    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(searchsorted)
```

```
/home/user6/miniconda3/lib/python3.10/site-packages/sklearn/base.py:420: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
0.9998642696691528
0.9998643504496011
[4.186060400407193, 0.9998642696691528, 0.9998643504496011, 0.015147822162040171, 1.6910202170427797, 2.85954937444741, 0.0010749909220781595, 0.09576922536595336, 0.9997021149836163]
The leaf part tested is unhealthy
         58304 function calls (57747 primitive calls) in 9.544 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
      196    0.000    0.000    0.001    0.000 <__array_function__ internals>:177(all)
        5    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(any)
       98    0.000    0.000    0.001    0.000 <__array_function__ internals>:177(array_equal)
      197    0.000    0.000    0.001    0.000 <__array_function__ internals>:177(atleast_1d)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(atleast_2d)
       11    0.000    0.000    0.008    0.001 <__array_function__ internals>:177(bincount)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(can_cast)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(clip)
      197    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(copy)
       21    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(copyto)
       12    0.000    0.000    0.017    0.001 <__array_function__ internals>:177(cumsum)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(delete)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(einsum)
      294    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(empty_like)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(expand_dims)
        2    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(in1d)
        2    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(linspace)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(may_share_memory)
       10    0.000    0.000    0.002    0.000 <__array_function__ internals>:177(mean)
        8    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(ndim)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(ptp)
        5    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(reshape)
        5    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(result_type)
        2    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(searchsorted)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(setdiff1d)
       88    0.000    0.000    0.004    0.000 <__array_function__ internals>:177(sum)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(take)
```



Conclusion:

1. A total of 5 threads are required for the code, 4 threads for splitted leaf images and 1 master thread for combining and displaying the final result
2. From the results received we can conclude that the average speed up time was 1.54ms for 2 VMS
3. Therefore, the lead disease detection on Cluster was found to be **faster** than on local machine.
4. Thus, the performance on Cluster for leaf disease detection was found to be better as compared to local machine
5. After comparing the disease detection accuracy with different Machine Learning models, it was found that **Logistic Regression** performed the best with highest accuracy of **96.2%**
6. We could successfully detect and classify the examined diseases with a accuracy of around **87.6% in average**. The **minimum accuracy value** was **76.3%.**
7. It is possible to further enhance the results and efficiency on a large scale for detecting diseases in thousands of leaves by increasing the number of nodes to hundreds and making the cluster larger
8. The only limitation of my project was that it was not possible to create a large network of nodes for clusters since here in my project I have used only Cluster.

References:

1. Mohanty SP, Hughes DP and Salathé M (2014) Using Deep Learning for Image-Based Plant Disease Detection. Front. Plant Sci.

2. Detection of plant leaf diseases using image segmentation and soft computing techniques (2016). Information Processing in Agriculture

3. Dheeb Al Bashish, Malik Braik and Sulieman Bani-Ahmad (2011). Detection and Classification of Leaf Diseases using K-means-based Segmentation and Neural-networks-based Classification. Information Technology Journal, 10: 267-275

4. Radha, Suja. (2017). Leaf Disease Detection using Image Processing. Journal of Chemical and Pharmaceutical Sciences

5. Guiling Sun, Xinglong Jia, and Tianyu Geng. Plant Diseases Recognition Based on Image Processing Technology (2018). Journal of Electrical and Computer Engineering

6. Manu BN (2020). Plant Leaf Disease Detection and Classification using Multiclass SVM Classifier, MATLAB Central File Exchange.

7. Saradhambal.G, Dhivya.R, Latha.S, R. Rajesh (2018). Plant Disease Detection and Its Solution Using Image Classification. International Journal of Pure and Applied Mathematics

8. Sushil R. Kamlapurkar (2015). Detection of Plant Leaf Disease Using Image Processing Approach. International Journal of Scientific and Research Publications

9. Sladojevic S, Arsenovic M, Anderla A, Culibrk D, Stefanovic D (2016). Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification. Comput Intell Neurosci.

10. Kumar KV and Jayasankar T. (2019) An identification of crop disease using image segmentation. Int J Pharm Sci & Res