MEDICINAL PLANT IDENTIFICATION USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

MUGESH. M	-	**820420106032**
MAMANNAN. K	-	**820420106025**
MOHAN. S. G	-	**820420106031**
DHASARATHAN. D	-	**820420106013**

In partial fulfilment for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

ANJALAI AMMAL MAHALINGAM ENGINEERING COLLEGE

KOVILVENNI – 614 403

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2024

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report " **MEDICINAL PLANT IDENTIFICATION USING MACHINE LEARNING**" is the bonafide work of "**MUGESH. M [820420106032], MAMANNAN. K [820420106025], MOHAN. S. G [820420106031] , DHASARATHAN. D [820420106013]**" who carried out the project work under my supervision.

SIGNATURE

Dr. A. Bhavani Sankar, M.E., Ph.D.,

HEAD OF THE DEPARTMENT

DEPARTMENT OF ECE,

Anjalai Ammal Mahalingam Engineering College,

Kovilvenni – 614 403,

Thiruvarur (DT)

SIGNATURE

Dr. A. Bhavani Sankar, M.E., Ph.D.,

SUPERVISOR

HEAD OF THE DEPARTMENT

DEPARTMENT OF ECE,

Anjalai Ammal Mahalingam Engineering College,

Kovilvenni – 614 403,

Examined on : .....

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

Medicinal plants have been used for centuries in traditional medicine systems for their therapeutic properties. Accurate identification of these plants is essential for ensuring their safe and effective use. In this project, we propose a machine learning-based approach for the automated identification of medicinal plants from images. We utilize a dataset of images containing various medicinal plant species and employ convolutional neural networks (CNNs) to extract features and classify plants into their respective species. Additionally, we explore the use of transfer learning to leverage pretrained CNN models for improved classification performance. Our results demonstrate the effectiveness of the proposed approach in accurately identifying medicinal plants, with potential applications in healthcare, biodiversity conservation, and traditional knowledge preservation. The accuracy of the project was 96 percent. This project highlights the synergy between machine learning and botanical sciences, showcasing how advanced technologies can enhance our understanding and utilization of medicinal plants for the benefit of society.

TABLE OF CONTENT

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	**I**
	LIST OF FIGURES	**IV**
1.	**INTRODUCTION**	1
	1.1 IMAGE PROCESSING	1
	1.2 TYPES OF IMAGE PROCESSING	2
	1.3 LEAF DETECTION AND PREDICTION	4
	1.4 LITERATURE REVIEW	6
2.	**SYSTEM ANALYSIS**	**11**
	2.1 PROBLEM DESCRIPTION	11
	2.2 EXISTING SYSTEM	11
	2.3 DESCRIPTION OF THE SYSTEM	12
	2.4 FEATURES EXTRACTION	13
3.	**PROPOSED SYSTEM**	**15**
	3.1 INTRODUCTION	15
	3.2 SYSTEM ARCHITECTURE	16
	3.3 MODULES DESCRIPTION	17
	3.3.1 Image Acqusition	17
	3.3.2 Preprocessing	17
	3.3.3 Image Segmentation	18
	3.3.4 Detection Prediction	19
	3.3.5 Evaluation Criteria	19
	3.4 SYSTEM SPECIFICATION	28
	3.4.1 Hardware Requirements	28
	3.4.2 Software Requirements	28

4.	**SOFTWARE DESCRIPTION**	**29**
	4.1 INTRODUCTION – MATLAB	29
	4.1.1 Toolboxes	30
	4.2 THE MATLAB SYSTEM	30
	4.2.1 Handle Graphics	30
	4.3 MATLAB FUNCTION	31
	4.4 MATLAB APPLICATION	33
	4.5 CONVOLUTION NEURAL NETWORK (CNN)	35
	4.5.1 Layers of CNN	35
	4.5.2 Types of CNN	36
5.	**RESULT AND CONCLUSION**	**39**
	5.1 SOURCE CODE	40
	5.2 RESULT	58
6.	**FUTURE ENHANCEMENT**	**61**
7.	**REFERENCE PAPERS**	**62**

LIST OF FIGURES

FIG NO	TITLE	PAGE NO
1.1	Basic Image Processing Steps	3
3.2	System Architecture	16
3.3	Level 0	21
3.4	Level 1	22
3.5	Level 2	23
3.6	UML Diagram	24
3.7	Class Diagram	25
3.8	Sequence Diagram	26
3.9	Activity Diagram	27
4.4	Simulink	34
5.2.1	Input Image	58
5.2.2	Preprocessing	58
5.2.3	Segmentation	59
5.2.4	Architecture	59
5.2.5	Evaluating	60
5.2.6	Classified Output	60
5.2.7	Accuracy	61

CHAPTER 1

1. INTRODUCTION

1.1 IMAGE PROCESSING

Image processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image. Usually **Image Processing** system includes treating images as two dimensional signals while applying already set signal processing methods to them.

It is among rapidly growing technologies today, with its applications in various aspects of a business. [13] Image Processing forms core research area within engineering and computer science disciplines too.

Image processing basically includes the following three steps.

1. Importing the image with optical scanner or by digital photography.

2. Analyzing and manipulating the image which includes data compression and image enhancement and spotting patterns that are not to human eyes like satellite photographs.

3. Output is the last stage in which result can be altered image or report that is based on image analysis.

Purpose of Image processing

The purpose of image processing is divided into 3 groups. They are:

1. Visualization - Observe the objects that are not visible.
2. Image sharpening and restoration - To create a better image.
3. Image retrieval - Seek for the image of interest.
4. Measurement of pattern – Measures various objects in an image.
3. Image Recognition – Distinguish the objects in an image.

1.2 TYPES

The two types of **methods used for Image Processing** are **Analog and Digital** Image Processing. Analog or visual techniques of image processing can be used for the hard copies like printouts and photographs. Image analysts use various fundamentals of interpretation while using these visual techniques. The image processing is not just confined to area that has to be studied but on knowledge of analyst. Association is another important tool in image processing through visual techniques. So analysts apply a combination of personal knowledge and collateral data to image processing.

Digital Processing techniques help in manipulation of the digital images by using computers. As raw data from imaging sensors from satellite platform contains deficiencies. To get over such flaws and to get originality of information, it has to undergo various phases of processing. The three general phases that all types of data have to undergo while using digital technique are Pre- processing, enhancement and display, information extraction.

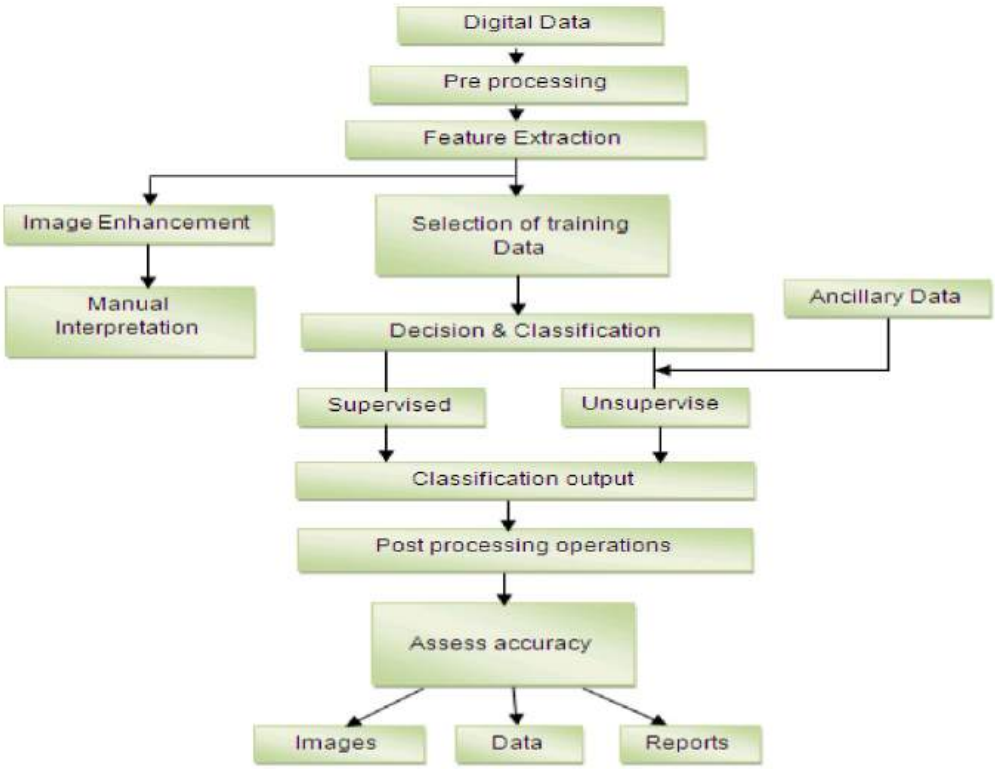

Fig 1.1 Basic image processing steps

In recent years, advances in information technology and telecommunications have acted as catalysts for significant developments in the sector of health care. These technological advances have had a particularly strong impact in the field of medical imaging, where film radiographic techniques are gradually being replaced by digital imaging techniques, and this has provided an impetus to the development of integrated hospital information systems and integrated tele radiology services networks which support the digital transmission, storage, retrieval, analysis, and interpretation of distributed multimedia patient records. One of the many added-value services that can be provided over an integrated tele radiology services network is access to high-performance computing facilities in order to execute computationally intensive image analysis and visualization tasks. In general, currently available products in the field of image processing (IP) meet only specific needs of different end user groups. They either

aim to provide a comprehensive pool of ready to use software within a user-friendly and application specific interface for those users that use IP software, or aim for the specialized IP researcher and developer, offering programmer's libraries and visual language tools. However, we currently lack the common framework that will integrate all prior efforts and developments in the field and at the same time provide added-value features that support and in essence realize what we call a 'service'. In the case of image processing, these features include: computational resource management and intelligent execution scheduling; intelligent and customizable mechanisms for the description, management, and retrieval of image processing software modules; mechanisms for the "plug-and-play" integration of already existing heterogeneous software modules; easy access and user transparency in terms of software, hardware, and network technologies; sophisticated charging mechanisms based on quality of service; and, methods for the integration with other services available within an integrated health telematics network.

1.3 LEAF DETECTIONS PREDICTION:

India is an agricultural country. Farmers have wide range of diversity to select suitable fruit and vegetable crop. Research work develops the advance computing system to identify the detection using infected images of various leaf spots. Images are captured by digital camera mobile and processed using image growing, then the part of the leaf sport has been used for the classification purpose of the train and test [10]. The technique evolved into the system is both Image processing techniques and advance computing techniques.

Image Analysis Can Be Applied For The Following Purposes:

1. To detect detection leaf, stem, fruit.
2. To quantify affected area by detection.

3. To find the boundaries of the affected area.

4. To determine the color of the affected area.

5. To determine size & shape of leaf.

6. To identify the Object correctly. Etc.

Most leaf detection are caused by fungi, bacteria and viruses. Fungi are identified primarily from their morphology, with emphasis placed on their reproductive structures. Bacteria are considered more primitive than fungi and generally have simpler life cycles. With few exceptions, bacteria exist as single cells and increase in numbers by dividing into two cells during a process called binary fission viruses are extremely tiny particles consisting of protein and genetic material with no associated protein [9]. In biological science, sometimes thousands of images are generated in a single experiment. There images can be required for further studies like classifying lesion, scoring quantitative traits, calculating area eaten by insects, etc. Almost all of these tasks are processed manually or with distinct software packages. It is not only tremendous amount of work but also suffers from two major issues: excessive processing time and subjectiveness rising from different individuals. Hence to conduct high throughput experiments, plant biologist need efficient computer software to automatically extract and analyze significant content. Here image processing plays important role. This project provides image processing techniques used for studding leaf detection.

1.4 LITERATURE REVIEW

1.4.1 Plant species identification using Elliptic Fourier leaf shape analysis Author: Joao Camargo Neto

Plant species can be accurately identified using Elliptic Fourier shape features of whole, extracted leaflets from plant canopies. The EF method combined with principle component analysis and linear discriminate models performed very well. Older plants of the third week had more mature leaves and provided the best leaf images for identifying plant species, demonstrated with an 88.3% classification rate. As leaves develop, their shapes become an important species trade mark. The redroot pigweed plant had the lowest correctly identified rate during both weeks. Redroot pigweed is misclassified with soybean during the third week, because of a similar rounded leaf shape to the trifoliate soybean leaflet at this growth stage. Some lesser misclassification with sunflower also occurred. By combining the leaflet images for the second and third weeks, an overall species identification accuracy of approximately 88.4% was obtained. Future EF studies should consider improved timing, background lighting, improved camerawork and application to identifying compound leaves. Leaf orientation is important in the EF analysis. To our knowledge, this has not been treated in previous plant species imaging studies. Two angles are needed to describe a leaf plane in three-dimensional space. One of the leaf angles is hard to control or adapt to, but sunlit leaves of many species can present themselves heliotropically toward a light source, such that one could select the best camera angles for full leaf exposure at the top of the canopy. The leaf angle in the plane of the canopy is apparently taken care of by the first EF harmonic, and that is a critical angle for rotationally invariant leaf texture or venation analysis. Additional studies regarding leaf orientation relative to the camera lens might help to resolve classification errors. Future studies are also needed to determine

minimal digital image resolutions needed to maintain the highest species discrimination performance.

1.4.2 First steps toward an electronic field guide for plants

Author: Gaurav Agarwal

This paper aims to construct prototype electronic field guides for the plant species represented in the Smithsonian's collection in the United States National Herbarium (US). We consider three key components in creating these field guides. First, we are constructing a digital library that is recording image and textual information about each specimen. We initially concentrated our efforts on type specimens (approximately 83,000 specimens of vascular plants at US) because they are critical collections and represent unambiguous species identification. We envision that this digital collection at the Smithsonian can then be integrated and linked with the digitized type specimen collections at other major world herbaria, such as the New York Botanical Garden, the Missouri Botanical Garden, and the Harvard University Herbaria. However, we soon realized that in many cases the type specimens are not the best representatives of variation in a species, so we have broadened use of non-type collections as well in developing the image library. Second, we are developing plant recognition algorithms for comparing and ranking visual similarity in the recorded images of plant species. These recognition algorithms are central to searching the image portion of the digital collection of plant species and will be joined with conventional search strategies in the textual portion of the digital collection. Third, we are developing a set of prototype devices with mobile user interfaces to be tested and used in the field. In this paper, we will describe our progress towards building a digital collection of the Smithsonian's type specimens, developing recognition algorithms that can match an image of a leaf to the species of plant from which it comes, and designing user interfaces for electronic field guides.

1.4.3 Automatic Leaf Detetction Classification for a Mobile Field Guide

Author: David Knight

Historically, identifying an unknown plant species required the consultation of a heavy field guide, where the user had to make sometimes obscure observations of the plant's features and navigate a complex decision tree. The process was nontrivial even for seasoned botanists, and carrying the guides into the field was often impractical – particularly for hikers and other casual users. The advent of highly-portable, computationally-powerful smart phones with large storage capacity presents an opportunity to not only replace and improve the database and decision tree functions of the field guide, but also to create automatic leaf classification applications based on well-known image processing methods currently becoming standardized on mobile platforms. A user friendly application on a popular mobile platform such as Android that is capable of identifying a good number of plant species could achieve widespread adoption, increasing the public's knowledge of and appreciation for their environment. The problem is complicated in this application by complex backgrounds that make image segmentation difficult, but simplified by foreknowledge of one or two desirable crops amongst unwanted weed species. Color and texture information is often sufficient to make this distinction, as opposed to more general applications, where numerous shape features must be acquired. More recently, several groups have approached the problem of automatic leaf classification. Though the groups often use similar digital morphological features, e.g. rectangularity, sphericity, eccentricity, etc., there is great variation in how these measures are combined and used in classification.

1.4.4 Leaf shape based plant species recognition

Author: Ji-Xiang Du

Plants can be usually identified according to the shapes, colors, textures and structures of their leaf, bark, flower, seedling and morph. However, it is very difficult for ones to analyze the shapes of flowers, seedling and morph of plants for their complex 3D structures if based on only 2D images. So in our research work, we will identify different plants by leaf features. Leaves are usually firstly clustered so that it is not easy for us to automatically extract features of leaves from the complex background. The leaf image database used in the following experiment is collected and built by ourselves in our lab. The procedure is that we pluck the leaf from plant, put it on the scanner, and then take the digital color image of the leaf directly, or put it on a panel, take digital color image of the leaf with a digital camera. In this way, we can get an image including only one leaf, and the background of the leaf image will be blurred. In this paper, a digital morphological feature based automatic recognition method for plant images was proposed and performed. The data usually contain noises, which result in overlapping samples in pattern space, and there may produce some outliers in the training data set. So we need to remove these outliers from the training data set so that a better decision boundary can be easily formed. The fifteen features are used to classify 20 species of plant leaves. In addition, a new moving median centers hyper sphere classifier is adopted to perform the classification. The experimental results demonstrated that the proposed method is effective and efficient. In particular, by comparing with the 1-NN and k-NN classifiers, it can be found that the MMC classifier can not only save the storage space but also reduce the classification time under the case of no sacrificing the classification accuracy.

1.4.5 Designing a mobile user interface for automated species identification

Author: Sean White

Human interaction is required because the vision algorithms are not perfect. The user can pan and zoom to inspect individual virtual vouchers and compare them with the plant sample. Semantic zooming is accomplished by either tapping on a virtual voucher to zoom in a level and reveal sets of voucher images, identification information, and textual descriptions, or by dragging up or down for continuous zooming. Once the identification has been verified by the botanist, a button press associates the identified species with the sample. A zoom able history of samples can be browsed to recall prior samples and search results. In this paper, we present a new Tablet-PC–based prototype, Leaf-View, which is being field-tested by our botanist colleagues at the Smithsonian Institution, and discuss design decisions based on user feedback and observations. The contribution focuses on interaction and use specific to automated identification in the field. Issues we address include inspection and comparison, immediate and batch processing, feedback from the identification process (via segmentation) as first steps to interacting with a vision algorithm, and the incorporation of identification in the collection process in contrast with post hoc identification. We plan to investigate other ways to interact with the vision algorithms as part of that research. We are also developing web-based and cell-phone–based user interfaces for identification so that non-specialists have access to the same tools. We hope that wider access will increase curiosity, understanding, and appreciation for the natural world.

CHAPTER 2
2. SYSTEM ANALYSIS

2.1 PROBLEM DESCRIPTION

Develop a robust machine learning system capable of accurately identifying medicinal plants from images. With the increasing popularity of alternative medicine and the importance of preserving traditional medicinal knowledge, there is a growing need for automated systems that can assist in the identification of medicinal plants. However, the vast diversity of plant species, variations within species, and the complexity of plant morphology pose significant challenges for accurate identification. The goal of this project is to create a solution that can reliably classify images of medicinal plants into their respective species or categories. The system should be able to handle variations in image quality, lighting conditions, and background clutter commonly encountered in real-world scenarios. Additionally, it should provide interpretable results, allowing users to understand the basis for classification decisions and potentially contribute to the collective knowledge of medicinal plants.

2.2 EXISTING SYSTEM

2.2.1 Introduction

Precision Botany (PB) refers to the application of new technologies in plant identification. Computer vision can be used in PB to distinguish plants from its species level, so that an identification can be applied on the size and number of plants detected for the classification purpose. Automatic plant identification tasks have gained recent popularity due to its use in quick characterization of plant species without requiring the expertise of botanists. Leaf-based features are preferred over flowers, fruits, etc. due to the seasonal nature of the later and also the abundance of leaves (except may be for the winter season). The current

electronic devices for capturing images have been developed to a point where there is little or no difference between the target and its digital counterpart. The success of machine learning for image recognition also suggests applications in the area of identification of plant by herbarium specimens. Once the image of a target is captured digitally, a myriad of image processing algorithms can be used to extract features from it. The use of each of these features will depend on the particular patterns to be highlighted in the image [14]. The automatic classification by computer vision of plants has received increasing attention in the recent past. Historically, identifying an unknown plant species required the consultation of a heavy field guide, where the user had to make sometimes obscure observations of the plant's features and navigate a complex decision tree. The process was nontrivial even for seasoned botanists, and carrying the guides into the field was often impractical – particularly for hikers and other casual users. The advent of highly-portable, computationally-powerful smart phones with large storage capacity presents an opportunity to not only replace and improve the database and decision tree functions of the field guide, but also to create automatic leaf classification applications based on well-known image processing methods currently becoming standardized platforms.

2.3 DESCRIPTION OF THE SYSTEM

Plant recognition he machine learning based classification of medical plant leaves. The total six varieties of medicinal plant leaves-based dataset are collected from the Department of Agriculture, These plants are commonly named in English as (herbal) Tulsi, Peppermint, Bael, Lemon balm, Catnip, and Stevia and scientifically named in Latin as Ocimum sanctum, Mentha balsamea, Aegle marmelos, Melissa officinal is, Nepeta cataria, and Stevia rebaudiana, respectively. The multispectral and digital image dataset are collected via a computer vision laboratory setup. This was referred from MPInet: Medicinal

Plants Identification using Deep Learning Preethi Salian K;Shrisha H.S.;Supriya Salian;Karthik K 2023 7th International Conference on Electronics, Communication and Aerospace Technology (ICECA).For the preprocessing step, we crop the region of the leaf and transform it into a gray level format. Secondly, we perform a seed intensity-based edge/line detection utilizing Sobel filter and draw five regions of observations. A total of 63 fused features dataset is extracted, being a combination of texture, run-length matrix, and multi-spectral features. For the feature optimization process, we employ a chi-square feature selection approach and select 14 optimized features. Finally, five machine learning classifiers named as a multi-layer perceptron, logit-boost, bagging, random forest, and simple logistic are deployed on an optimized medicinal plant leaves dataset, and it is observed that the multi-layer perceptron classifier shows a relatively promising accuracy of 99.01% as compared to the competition. The distinct classification accuracy by the multi-layer perceptron classifier on six medicinal plant leaves are 99.10% for.

2.4 FEATURES EXTRACTION

2.4.1 Leaf Shape features

Shape features consist, generally speaking, of a set of measurements that describe a certain shape according to some of its fundamental geometric properties. Commonly used features for the description of a certain shape are for example its aspect ratio, rectangularity, circularity, solidity, compactness and convexity. This was referred from Identification and classification of medicinal plants of the Indian Himalayan region using Hyperspectral remote sensing and random forest techniques. The nomenclatures may vary considerably but the general definition of these measures is very consistent and they are presented in any introductory book on image processing, such as for instance [12]. Shape features are intensively used in image processing and many times not necessarily in

classification contexts. One of the most interesting aspects of shape analysis through shape features is that this technique is very intuitive, i.e., shape features are easy to understand as they quantify basic geometric properties which match the human visual perception. Even if a thorough description of the calculation process of some of the already mentioned shape features would not have been provided, the reader could easily guess what is meant for instances under a feature's name like for example rectangularity.

2.4.2 Leaf texture analysis

Leaf texture analysis consists of the application of common image texture analysis techniques to leaf images in plant classification systems [8]. This can be of use if texture exhibits consistent properties within a certain species and especially if used in combination with other leaf analysis methods like leaf shape techniques. Several authors have reported good results of the inclusion of texture analysis techniques in leaf classification systems, but given the different sizes of the databases employed as well as the nature of the employed methods, no consistent results about the quality of this approach are available

CHAPTER 3

3. PROPOSED SYSTEM

3.1 INTRODUCTION

Leaf detections The present work shows a comparison of classification of seven different representations of plant leaves using three features extracted from the image The proposed strategy utilizes Convolution Neural Network (CNN) that is There are many different types of herbal remedies and they can vary from place to place, resulting in a similar pattern of "size" and "shapes" . These plants have excellent medicinal properties from roots to leaves. The leaves of some herbs such as Karpooravalli (Coleus ambonicus), Podina (Mentha arvensis), Neem (Adidirachta indica), Thudhuvalai (Solanum trilobatum), Basil (Ocimum sanctum), etc., are used in our life today. Some leaves have their own medicinal properties such as skin diseases, colds, blood purifier, and indigestion. Thus, medical plants are plants used for their particular properties beneficial to human health, even animal health. First called "simple" from the Middle Ages in medieval medicine, today they correspond to products from traditional or modern herbal medicine. The plant is rarely used whole; at least one of its parts (leaf, stem, root, etc.) can be used to heal itself. Different parts of the same plant can have different uses

3.2 SYSTEM ARCHITECTURE

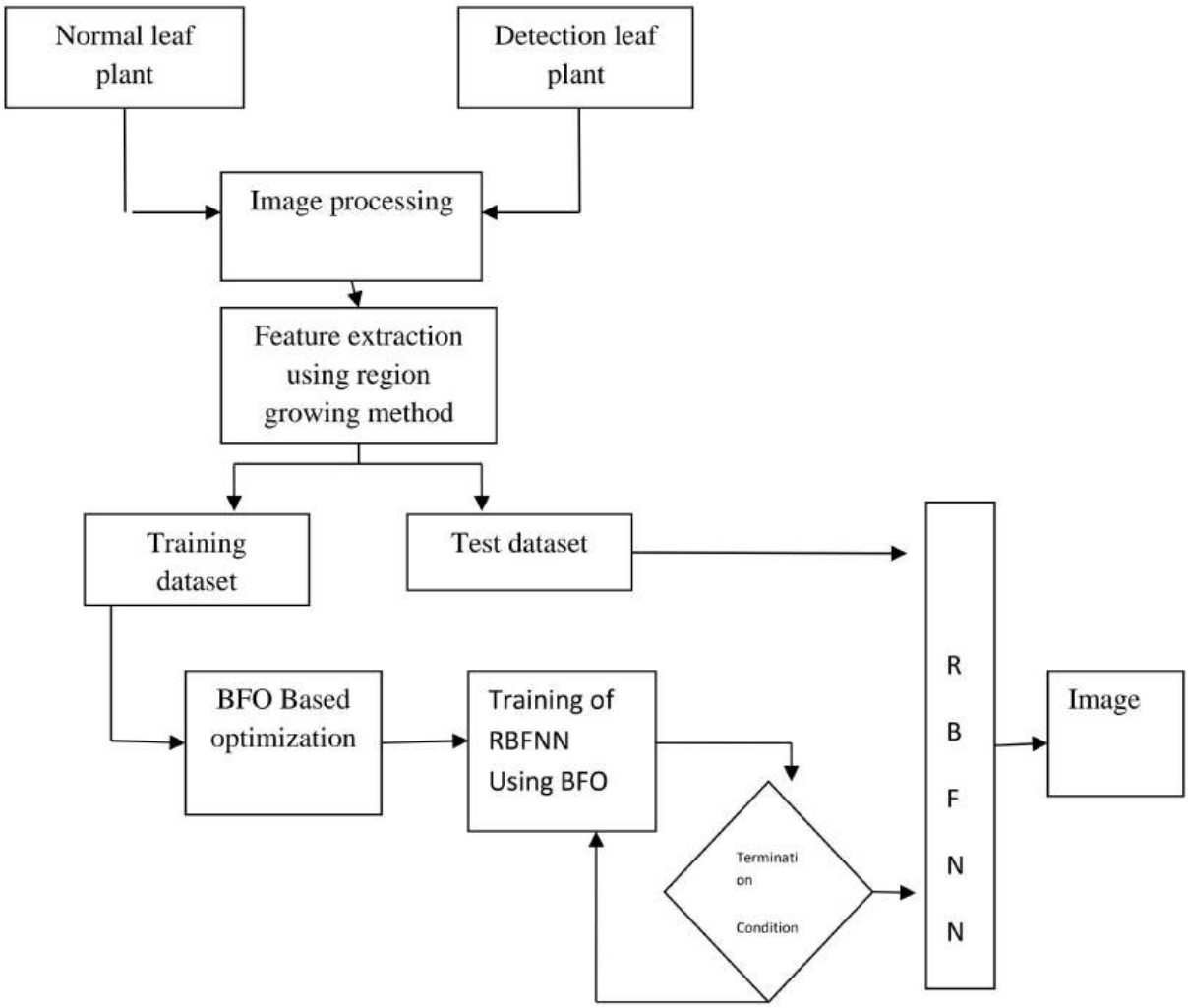

Fig 3.2 System Architecture

3.3 MODULES DESCRIPTION

- Image acquisition
- Preprocessing
- Image segmentation
- Detection prediction
- Evaluation criteria

3.3.1 Image Acquisition

Plants have become an important source of energy, and are a fundamental piece in the puzzle to solve the problem of global warming. There are several detection that affect plants with the potential to cause devastating economical, social and ecological losses. In this context, diagnosing detection in an accurate and timely way is of the utmost importance. There are several ways to detect plant pathologies. Some diseases do not have any visible symptoms associated, or those appear only when it is too late to act. In those cases, normally some kind of sophisticated analysis, usually by means of powerful microscopes, is necessary. In other cases, the signs can only be detected in parts of the electromagnetic spectrum that are not visible to humans. A common approach in this case is the use of remote sensing techniques that explore multi and hyper spectral image captures. The methods that adopt this approach often employ digital image processing tools to achieve their goals.

3.3.2 Preprocessing

In this module, we can implement preprocessing techniques to convert RGB image to gray image and remove the noises from images. The goal of preprocessing is

- Enhance the visual appearance of images.
- Improve the manipulation of datasets.

Using image resampling to reduce or increase the number of pixels of the dataset and improve the visualization by brightening the dataset. The first step in this process is to convert the acquired color image to a grayscale image. Following this, image segmentation is performed to identify leaf pixels and background pixels. After holes have been closed and small regions removed, the segmented image is converted to binary and the interior of the leaf is subtracted, leaving an image of the leaf's outline contour.

3.3.3 Image Segmentation

RGA is a simple approach that starts with the set of seed points and grows by using these seed points forming a region by appending to each seed the adjoining pixels, having analogous features to the seed such as intensity level, color, or scalar properties for the grayscale images. RGA method delivers the benefit of choosing several measures for selecting a seed point. There are two basic schemes for this technique termed as 4-neighborhood and 8-neighborhood. The 4-neighborhood leaving diagonally associated regions selects adjacent regions while 8-neighborhood selects both diagonal regions and adjacent regions while growing procedure.

B. BACTERIAL FORAGING OPTIMIZATION (BFO) FOR TRAINING THE NETWORK

BFO is new nature-inspired optimization algorithms proposed by Kevin Passino in 2002. The group foraging behavior of bacteria such as M. Xanthus and E. Coli. motivated the development of BFO. BFO algorithm is inspired by the chemotaxis behavior of virtual bacteria that move towards (in the direction of) or away (not in the direction of) from the specific signals taking small steps while searching for nutrients in the problem search space is another key concept for BFO. BFO has turned out to be an effective and influential optimization tool that provides

high convergence speed and accuracy applied in the number of the real world applications. The proposed method has been trained by using the four basic steps of BFO are given below:

3.3.4 Detection Prediction

CNN are a set of related supervised learning methods used for classification and regression. Supervised learning involves analyzing a given set of labeled observations (the training set) so as to predict the labels of unlabelled future data (the test set). Specifically, the goal is to learn some function that describes the relationship between observations and their labels. Multiclass CNN aims to assign labels to instances by using support vector machines, where the labels are drawn from a finite set of several elements. The dominant approach for doing so is to reduce the single multiclass problem into multiple binary classification problems. Common methods for such reduction include: building binary classifiers which distinguish between (i) one of the labels and the rest (one-versus-all) or (ii) between every pair of classes (one-versus-one). Classification of new instances for the one-versus-all case is done by a winner-takes-all strategy, in which the classifier with the highest output function assigns the class. Based on the multiclass classifier, we can predict diseases in leaf images.

3.3.5 Evaluation Criteria

It was implemented on MATLAB 2012b working on a system with an i3 processor having 4GB RAM. For validating the effectiveness of this work we have taken two sets of images. The first set consisted of 6 different images with 6 different detection is selected from planet natural and second dataset consists of about 270 images are selected from crowdAI.org (PlantVillage Disease Classification Challenge) categorizing among the same 6 set of detection. The result part is divided into two categories (A) To correctly segment/identify the

infected area on leaf detection for a detection and (B) To classify the type of leaf detection. The performance evaluation of the proposed work for correctly identifying the affected area or detection on the leaf detection is evaluated using two quantitative evaluation parameters that are based on the statistical performance of the ground truth image and segmented image. The parameters are specificity and sensitivity Refer to (9) and (10). The most critical part is the classification of diseases based on some attributes associated with them. The performance of the proposed work for correctly classifying detection is done by using two entropy functions known as Validation evaluation partition coefficient Vpc and Validation evaluation partition entropy Vpe

$$Specificity=TNTN+FP$$

$$Sensitivity=TPTP+FN$$

where True Positive (TP) = no. of pixels exactly classified, False Positive (FP) = no. of pixels incorrectly classified, True Negative (TN) = no. of pixels exactly misclassified, and False Negative (FN) = no. of pixels incorrectly misclassified. The value of specificity and sensitivity lies between 0 and 1 when result is equal to 1 means perfect segmentation.

$$Vpc= \Sigma\Sigma uik2NNi=1Kk=1$$

$$Vpe= -\Sigma\Sigma uik\log (uik)=1Kk=1$$

where uik is the membership value of pixel i belonging to the k-th cluster, K is the number of clusters and N is the total number of image pixels. Both functions value lies between 0 and 1, when Vpc is high and Vpe is low, it implies the membership values are less in segmentation results and the tissues are classified correctly.

Data Flow Diagram

A two-dimensional diagram that explains how data is processed and transferred in a system. The graphical depiction identifies each source of data and how it interacts with other data sources to reach a common output. Individuals seeking to draft a data flow diagram must (1) identify external inputs and outputs, (2) determine how the inputs and outputs relate to each other, and (3) explain with graphics how these connections relate and what they result in. This type of diagram helps business development and design teams visualize how data is processed and identify or improve certain aspects.

Level 0

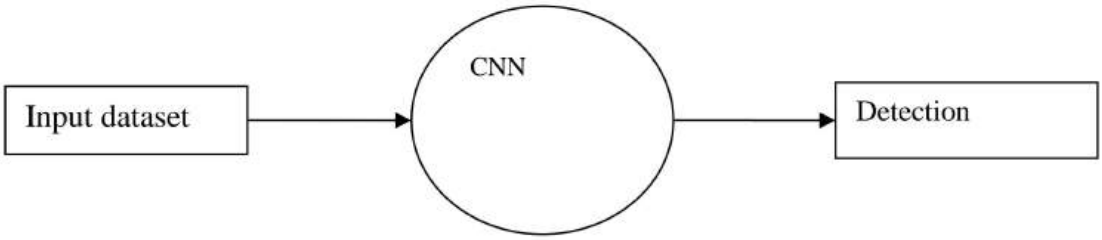

Fig 3.3 Level 0

DFD Level 0 is also called a Context Diagram. It's a basic overview of the whole system or process being analyzed or modeled. It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.

Level 1

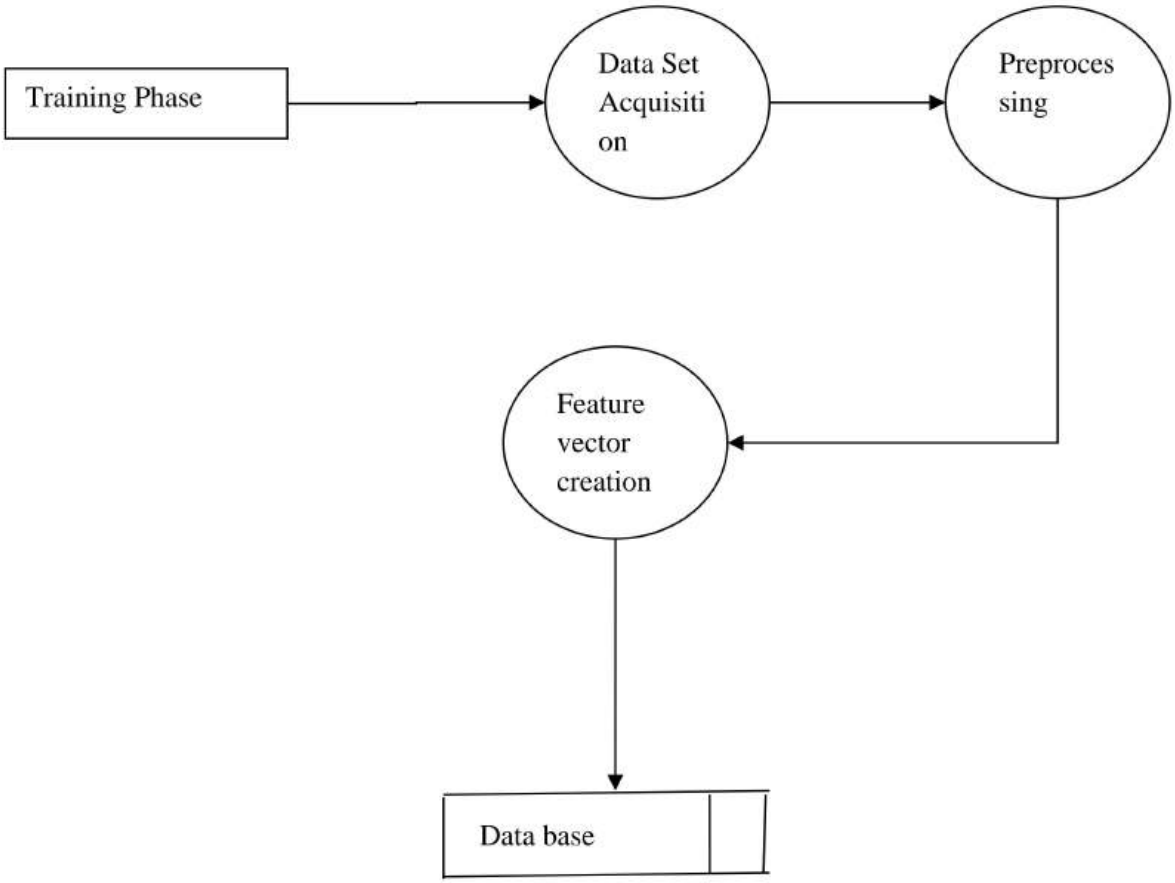

Fig 3.4 Level 1

DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram. It will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its sub processes.

Level 2

Prediction

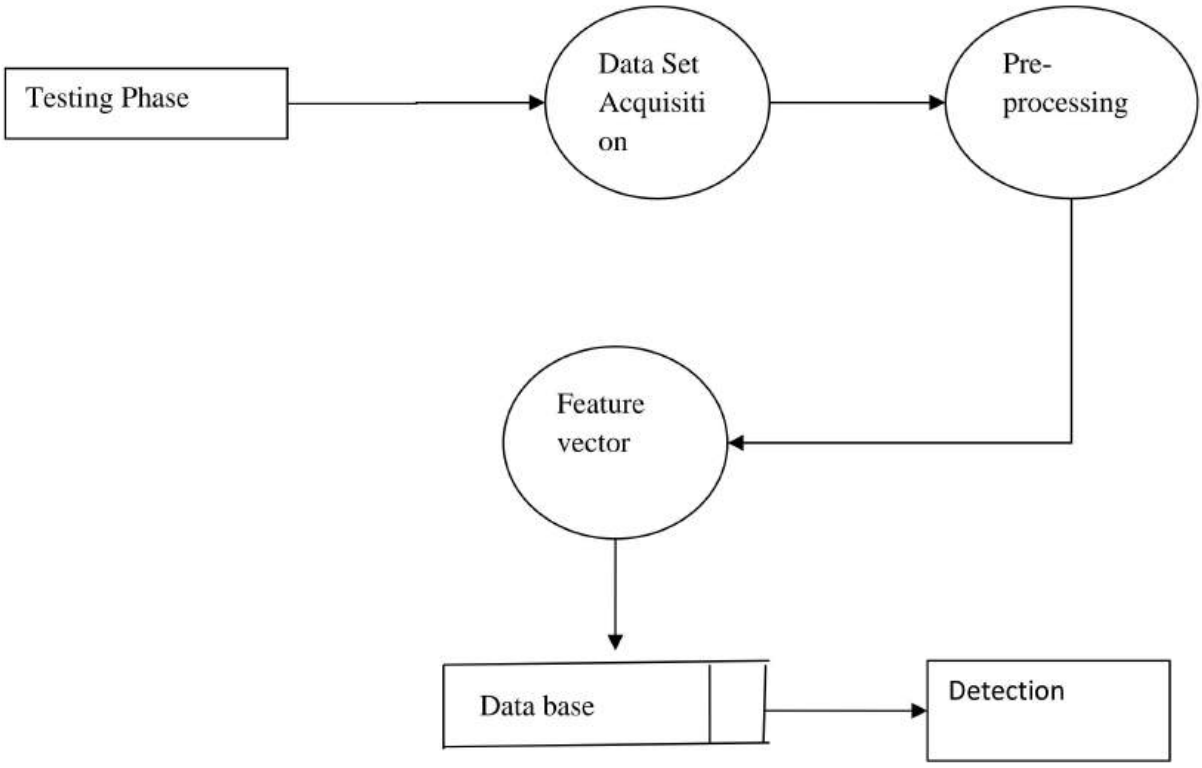

Fig 3.3 Level 2

DFD Level 2 then goes one step deeper into parts of Level 1. It may require more text to reach the necessary level of detail about the system's functioning.

UML DIAGRAMS

Usecase Diagram

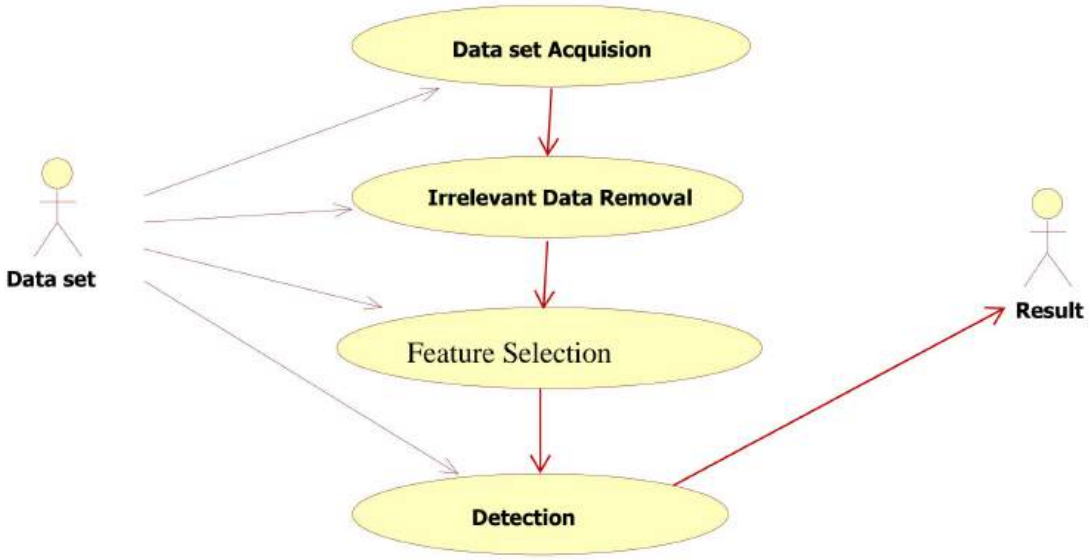

Fig 3.6 UML Diagrams

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

Class Diagram

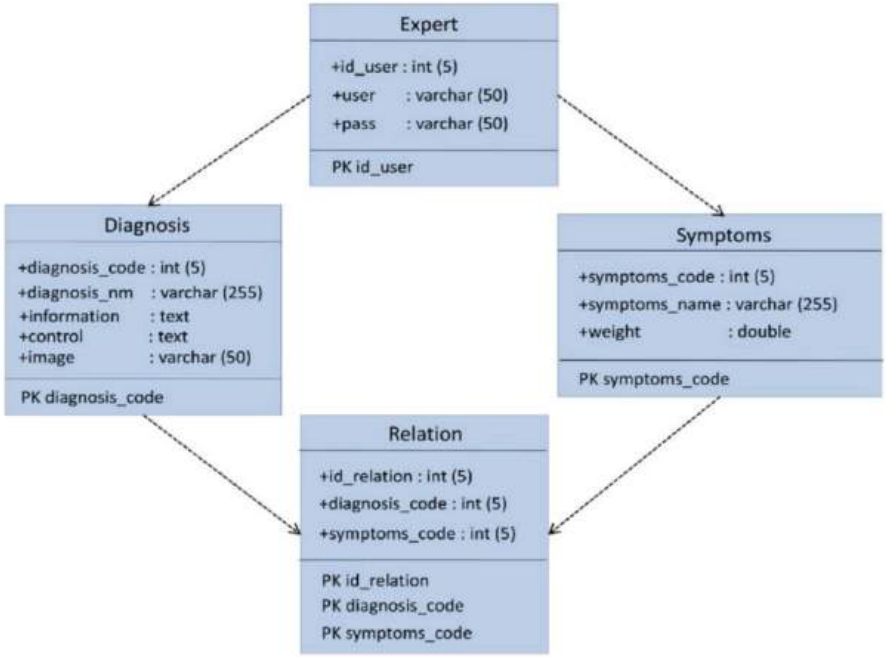

Fig 3.7 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

Sequence Diagram

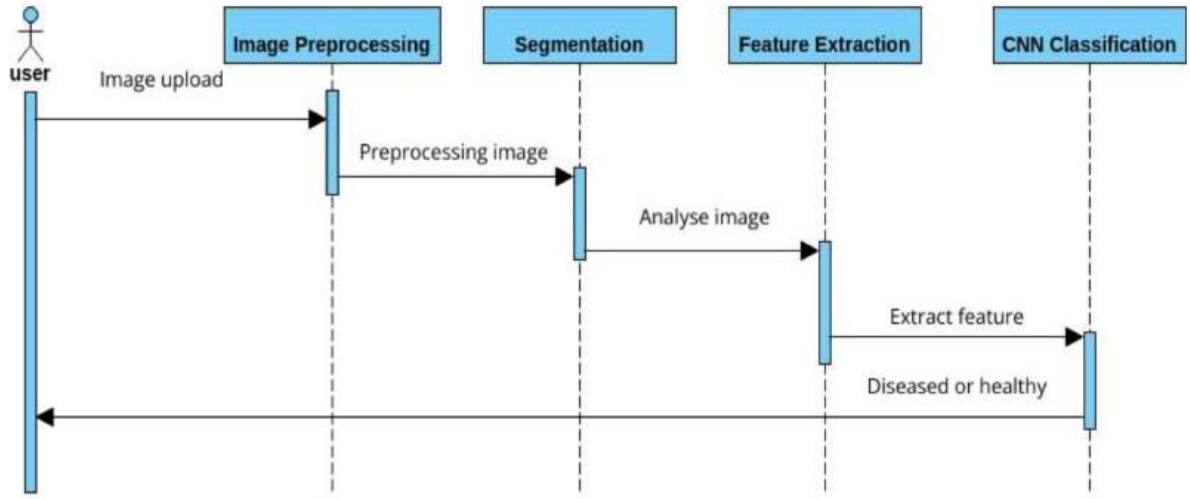

Fig 3.8 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development. Sequence diagrams are sometimes called event diagrams or event scenario a sequence diagram shows, as parallel vertical lines different processes or objects that live simultaneously and as horizontal arrows, messages exchanged between them.

Activity Diagram

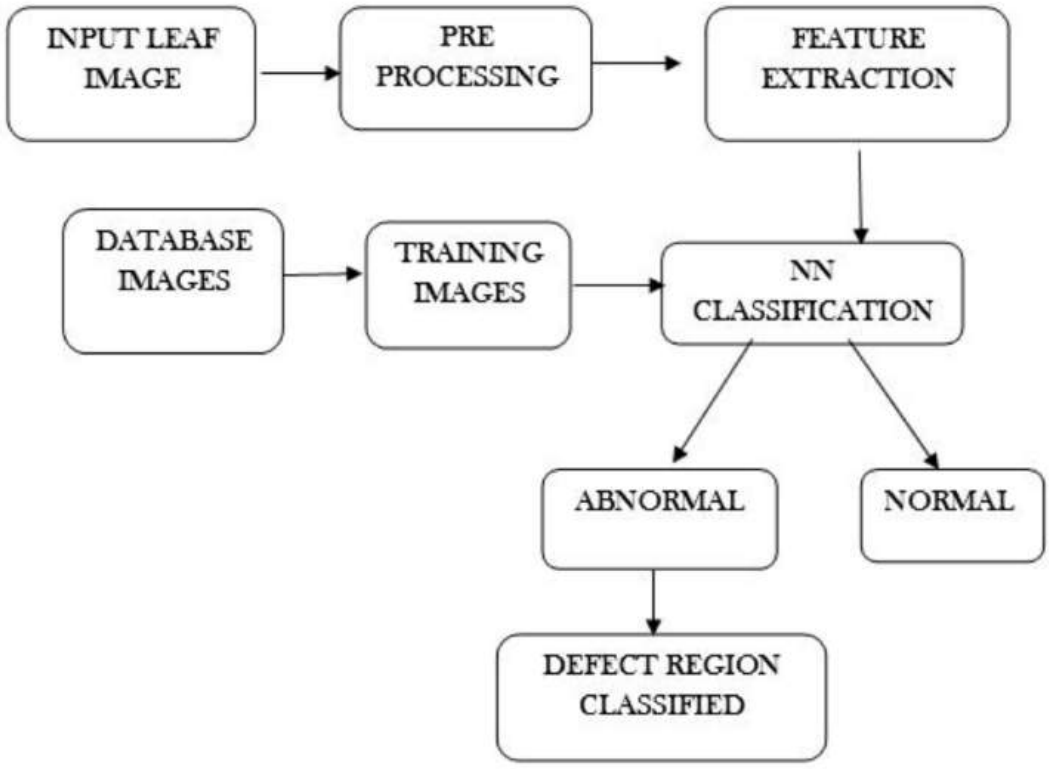

Fig 3.9 Activity Diagram

Activity diagram is a loosely defined diagram to show work flows of stepwise activities and actions with support for choice, iteration and concurrency. Activity diagrams may be regarded as a form of flowchart. Typical flowchart techniques lack constructs for expressing concurrency. However, the join and split symbols in activity diagrams only resolve this for simple cases; the meaning of the model is clear not when they are arbitrarily combined with decisions or loops

3.4 SYSTEM SPECIFICATION

- **3.4.1 Hardware Requirements**

• System	Intel 6.0.
• Hard Disk	230 GB.
• RAM	2 GB.
• Monitor	14" Color Monitor.
• Mouse	Optical Mouse.

- **3.4.2 Software Requirements**

• Operating system	Window 8(64bits).
• Front End	MAT LAB
• Database	Dataset.

CHAPTER 4

4. SOFTWARE DESCRIPTION

4.1 INTRODUCTION

MATLAB

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Math and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or Fortran.

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB uses software developed by the LAPACK and ARPACK projects, which together represent the state-of-the-art in software for matrix computation.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and

advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

4.1.1 Toolboxes

MATLAB features a family of application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

4.2 THE MATLAB SYSTEM

The MATLAB system consists of five main parts: Development Environment. This is the set of tools and facilities that help you use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, and browsers for viewing help, the workspace, files, and the search path.

The MATLAB Mathematical Function Library. This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

The MATLAB language: This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both “programming in the small” to rapidly create quick and dirty throw-away programs, and “programming in the large” to create complete large and complex application programs.

4.2.1 Handle Graphics

This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your

4.3 MATLAB FUNCTIONS:

A MATLAB "function" is a MATLAB program that performs a sequence of operations specified in a text file (called an m-file because it must be saved with a file extension of *.m). A function accepts one or more MATLAB variables as inputs, operates on them in some way, and then returns one or more MATLAB variables as outputs and may also generate plots, etc.

Some functions are

Imread() – Reading the image from the graphics file.

`A = imread(filename, fmt)` reads a grayscale or color image from the file specified by the string `filename`. If the file is not in the current folder, or in a folder on the MATLAB path, specify the full pathname.

The text string *`fmt`* specifies the format of the file by its standard file extension. For example, specify `'gif'` for Graphics Interchange Format files. To see a list of supported formats, with their file extensions, use the `imformats` function. If `imread` cannot find a file named `filename`, it looks for a file named `filename.fmt`.

The return value `A` is an array containing the image data. If the file contains a grayscale image, `A` is an M-by-N array. If the file contains a truecolor image, `A` is an M-by-N-by-3 array. For TIFF files containing color images that use the CMYK color space, `A` is an M-by-N-by-4 array. The class of `A` depends on the bits-per-sample of the image data, rounded to the next byte boundary. For

example, imread returns 24-bit color data as an array of uint8 data because the sample size for each color component is 8 bits.

[X, map] = imread(...) reads the indexed image in filename into X and its associated colormap into map. Colormap values in the image file are automatically rescaled into the range [0,1].

Image- write image to graphics file

imwrite(A,filename,*fmt*) writes the image A to the file specified by filename in the format specified by *fmt*.

A can be an M-by-N (grayscale image) or M-by-N-by-3 (truecolor image) array, but it cannot be an empty array. For TIFF files, A can be an M-by-N-by-4 array containing color data that uses the CMYK color space. For GIF files, A can be an M-by-N-by-1-by-P array containing grayscale or indexed images — RGB images are not supported. For information about the class of the input array and the output image. filename is a string that specifies the name of the output file. fmt can be any of the text strings listed. This list of supported formats is determined by the MATLAB image file format registry. See imformats for more information about this registry. imwrite(X,map,filename,*fmt*) writes the indexed image in X and its associated colormap map tofilename in the format specified by *fmt*. If X is of class uint8 or uint16, imwrite writes the actual values in the array to the file. If X is of class double, imwrite offsets the values in the array before writing, using uint8(X-1). map must be a valid MATLAB colormap. Note that most image file formats do not support colormaps with more than 236 entries. When writing multiframe GIF images, X should be an 4-dimensional M-by-N-by-1-by-P array, where P is the Text of frames to write. imwrite(...,filename) writes the image to filename, inferring the format to use from the filename's extension. imwrite(...,Param1,Val1,Param2,Val2...) specifies parameters that control various characteristics of the output file for HDF, JPEG, PBM, PGM,

PNG, PPM, and TIFF files. For example, if you are writing a JPEG file, you can specify the quality of the output image. For the lists of parameters available for each format.

4.4 MATLAB APPLICATION

The MATLAB Application Program Interface (API). This is a library that allows you to write C and Fortran programs that interact with MATLAB. It include facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

What Is Simulink?

Simulink, a companion program to MATLAB, is an interactive system for simulating nonlinear dynamic systems. It is a graphical mouse-driven program that allows you to model a system by drawing a block diagram on the screen and manipulating it dynamically. It can work with linear, nonlinear, continuous-time, discrete-time, multirate, and hybrid systems. Blocksets are add-ons to Simulink that provide additional libraries of blocks for specialized applications like communications, signal processing, and power systems.

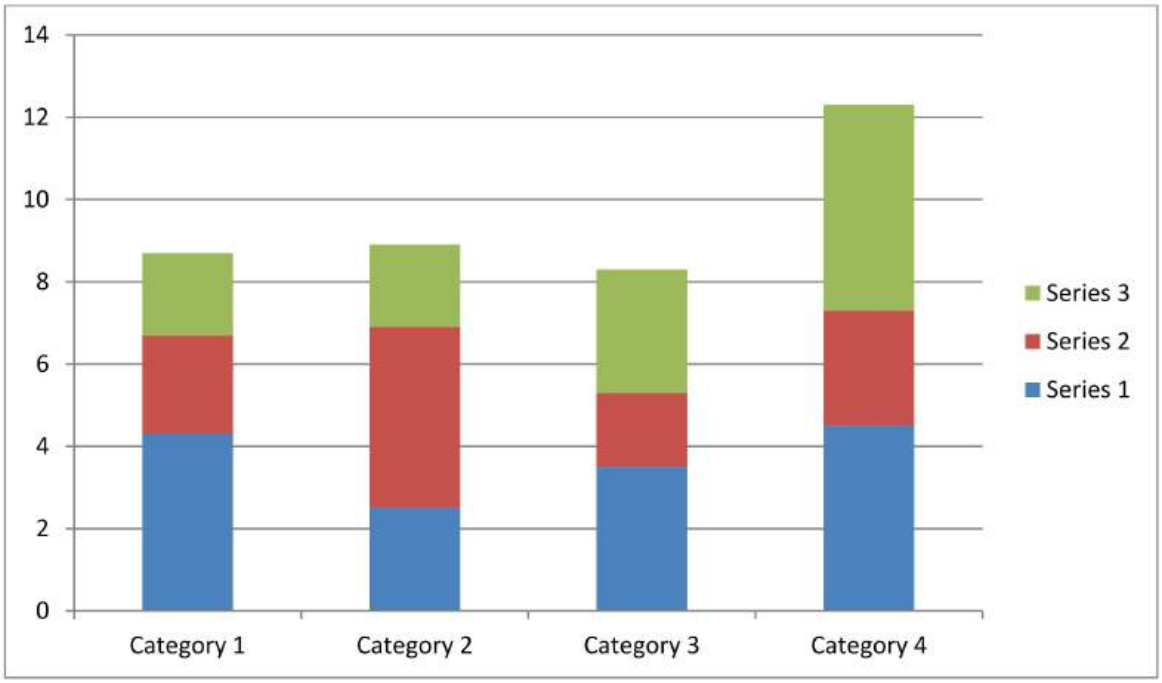

Fig 4.4 Simulink

Real-Time Workshop is a program that allows you to generate C code from your block diagrams and to run it on a variety of real-time systems.

Stateflow is an interactive design tool for modeling and simulating complex reactive systems. Tightly integrated with Simulink and MATLAB, Stateflow provides Simulink users with an elegant solution for designing embedded systems by giving them an efficient way to incorporate complex control and supervisory logic within their Simulink models. With Stateflow, you can quickly develop graphical models of event-driven systems using finite state machine theory, state chart formalisms, and flow diagram notation. Together, State flow and Simulink serve as an executable specification and virtual prototype of your system design.

4.5 CONVOLUTION NEURAL NETWORK (CNN)

CNN stands for Convolutional Neural Network, which is a type of deep learning model designed for processing and analyzing structured grid-like data, such as images or sequences. It is a specialized neural network architecture that has achieved remarkable success in computer vision tasks, including image classification, object detection, and image segmentation.

The fundamental idea behind CNNs is to leverage the concept of convolution, which is a mathematical operation that combines two functions to produce a third finction. In the context of CNNs, convolutional layers apply filters (also called kemels or feature detectors) to input data to extract meaningful features. These filters capture patterns and local relationships within the data, such as edges, textures, or shapes. The main advantages of CNNs are their ability to automatically learn hierarchical representations and their translation invariance properties. CNNs consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers. Convolutional layers perform the convolution operation, pooling layers reduce the spatial dimensions of the data,

and fully connected layers classify or predict the output based on the learned features.

4.5.1 Layers Of CNN :

Input Layer: This layer represents the input data, typically an image or a sequence of images. The input layer's dimensions correspond to the dimensions of the input data, such as the image width, height, and number of color channels.

Convolutional Layers: Convolutional layers apply filters or kernels to the input data, extracting local patterns and features. Each filter convolves across the input, computing dot products between the filter weights and the corresponding local patches of the input data. This process generates feature maps that highlight important spatial information in the input.

Activation Layers: Activation layers introduce non-linearity into the network by applying an activation function element-wise to the output of the convolutional layers. Common activation functions used in CNNs include Rectified Linear Unit (ReLU), sigmoid, and hyperbolic tangent.

Pooling Layers: Pooling layers down sample the feature maps obtained from the convolutional layers. They reduce the spatial dimensions of the feature maps while retaining important features. Max pooling and average pooling are the two popular pooling operations used in CNNs.

Dropout Layers: Dropout layers help prevent over fitting by randomly "dropping out" a fraction of the neurons during training. This regularization technique improves the generalization ability of the network and reduces the likelihood of over-reliance on specific features.

Fully Connected Layers: Fully connected layers, also known as dense layers, connect every neuron from the previous layer to the subsequent layer. These

layers take the high-level features extracted by the convolutional layers and perform classification or regression based on the learned representations.

Output Layer: The final layer of the CNN is the output layer, which provides the network's final predictions or outputs.

4.5.2 Types Of CNN:

1.LeNet-5: LeNet-S is one of the earliest CNN architectures developed by Yann LeCun et al. It was designed for handwritten digit recognition and consists of several convolutional and pooling layers, followed by fully connected layers. LeNet-5 served as the foundation for subsequent CNN architectures.

2. AlexNet: AlexNet, developed by Alex Krizhevsky et al., gained attention by winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. It has a deeper architecture compared to LeNet-5, utilizing multiple convolutional and pooling layers, and introducing the concept of ReLU activation. AlexNet played a crucial role in popularizing CNNs for image classification tasks.

3. VGGNet: The Visual Geometry Group Network (VGGNet) was developed by the Visual Geometry Group at the University of Oxford. VGGNet emphasizes network depth as a crucial factor in achieving better performance. It consists of a series of small-sized convolutional filters and deeper network architectures with up to 19 layers. VGGNet has been widely adopted and serves as a baseline for many subsequent CNN architectures.

4. GoogLeNet/Inception: GoogLeNet, also known as the Inception architecture, introduced the concept of inception modules. These modules use parallel convolutions of different sizes and concatenate their outputs to capture multi-scale features effectively. GoogLeNet's architecture helps improve both accuracy and computational efficiency by reducing the number of parameters.

5. ResNet: Residual Networks (ResNet) are known for their ability to tackle the vanishing gradient problem in very deep neural networks. ResNet introduces skip connections or shortcut connections that allow information to bypass several layers, facilitating the flow of gradients and enabling the training of extremely deep networks. ResNet architectures, such as ResNet-50, ResNet-101, and ResNet-152, have achieved state-of-the-art performance on various image recognition tasks.

6. MobileNet: MobileNet is designed for resource-constrained environments like mobile devices. It employs depth-wise separable convolutions, which separate

the spatial and channel-wise convolutions, reducing the number of parameters and computations required. MobileNet models are efficient and offer a good trade-off between accuracy and computational cost.

7. DenseNet: DenseNet is built on the idea of dense connections between layers. Each layer in a DenseNet receives direct connections from all preceding layers, resulting in highly connected feature maps. DenseNet enables efficient feature reuse and gradient flow, leading to better parameter efficiency and improved accuracy.

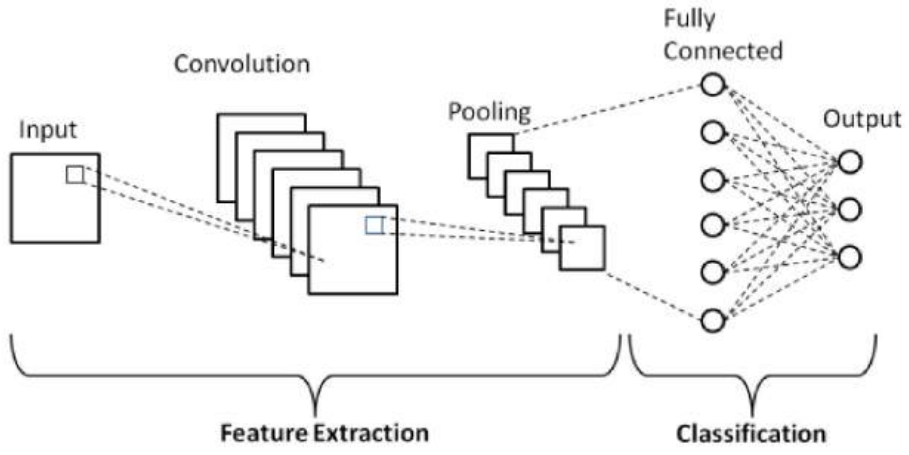

Fig 4.5 CNN Architecture

CHAPTER 5

5. RESULT AND CONCLUSION

The plant serves as the we develop a machine learning (ML) based medical plants leaves classification utilizing multispectral and texture dataset. The main objective is to collect a refined and standardized dataset, edge/line detection, fused features extraction, optimized extracted features, and select the most valuable feature and select the efficient ML classifiers. The fused (multispectral + texture) feature dataset holds six types of medicinal leaves named Tulsi, Peppermint, Bael, Lemon Balm, Catnip, and Stevia collected via computer vision laboratory setup. Due to the complex laboratory setup, the collected dataset is very refined and standardized. The chi-square feature selection approach provides the 14 most worthful features that are useful to obtain better classification results. The conclusion of medicinal plant identification using machine learning underscores its potential for revolutionizing traditional methods of plant recognition and classification. By harnessing the power of machine learning algorithms, researchers have achieved notable successes in automating the identification process, thereby significantly reducing time and labor costs. However, challenges such as data scarcity, model generalization, and interpretability persist, requiring further research and refinement. Despite these obstacles, the promising results suggest a bright future for leveraging machine learning in medicinal plant identification, facilitating expedited drug discovery, conservation efforts, and sustainable resource management.

5.1 SOURCE CODE:

Train :

```
Train_Feat=0;
for a1=1 : 38

I = imread([pwd,'\test\',num2str(a1),'.jpg']);
I = imresize(I,[236,236]);
a1
cform = makecform('srgb2lab');

lab_he = applycform(I,cform);

ab = double(lab_he(:,:,2:3));
nrows = size(ab,1);
ncols = size(ab,2);
ab = reshape(ab,nrows*ncols,2);
nColors = 3;
[cluster_idx cluster_center] = kmeans(ab,nColors,'distance','sqEuclidean',
...
                                      'Replicates',3);

pixel_labels = reshape(cluster_idx,nrows,ncols);

segmented_images = cell(1,3);

rgb_label = repmat(pixel_labels,[1,1,3]);

for k = 1:nColors
    colors = I;
    colors(rgb_label ~= k) = 0;
    segmented_images{k} = colors;
end

figure(1);
imshow(segmented_images{1});title(' Cluster 1 ');
figure(2);
imshow(segmented_images{2});title(' Cluster 2 ');
figure(3);
imshow(segmented_images{3});title(' Cluster 3 ');


% Feature Extraction
pause(3)
% x = inputdlg('Enter the cluster no. containing the ROI only:');
i = 3;
% Extract the features from the segmented image
seg_img =segmented_images{i};

% Convert to grayscale if image is RGB
if ndims(seg_img) == 3
   img = rgb2gray(seg_img);
end
%figure, imshow(img); title('Gray Scale Image');
```

```
% Evaluate the disease affected area
black = im2bw(seg_img,graythresh(seg_img));
%figure, imshow(black);title('Black & White Image');
m = size(seg_img,1);
n = size(seg_img,2);

zero_image = zeros(m,n);
%G = imoverlay(zero_image,seg_img,[1 0 0]);

cc = bwconncomp(seg_img,6);
diseasedata = regionprops(cc,'basic');
A1 = diseasedata.Area;
sprintf('Area of the disease affected region is : %g%',A1);

I_black = im2bw(I,graythresh(I));
kk = bwconncomp(I,6);
leafdata = regionprops(kk,'basic');
A2 = leafdata.Area;
sprintf(' Total leaf area is : %g%',A2);

%Affected_Area = 1-(A1/A2);
Affected_Area = (A1/A2);
if Affected_Area < 0.1
    Affected_Area = Affected_Area+0.13;
end
sprintf('Affected Area is: %g%%',(Affected_Area*100))
Affect = Affected_Area*100;
% Create the Gray Level Cooccurance Matrices (GLCMs)
glcms = graycomatrix(img);

% Derive Statistics from GLCM
stats = graycoprops(glcms,'Contrast Correlation Energy Homogeneity');
Contrast = stats.Contrast;
Correlation = stats.Correlation;
Energy = stats.Energy;
Homogeneity = stats.Homogeneity;
Mean = mean2(seg_img);
Standard_Deviation = std2(seg_img);
Entropy = entropy(seg_img);
RMS = mean2(rms(seg_img));
%Skewness = skewness(img)
Variance = mean2(var(double(seg_img)));
a = sum(double(seg_img(:)));
Smoothness = 1-(1/(1+a));
Kurtosis = kurtosis(double(seg_img(:)));
Skewness = skewness(double(seg_img(:)));
% Inverse Difference Movement
m = size(seg_img,1);
n = size(seg_img,2);
in_diff = 0;
for i = 1:m
    for j = 1:n
        temp = seg_img(i,j)./(1+(i-j).^2);
        in_diff = in_diff+temp;
    end
end
IDM = double(in_diff);
```

```
feat_disease = [Contrast,Correlation,Energy,Homogeneity, Mean,
Standard_Deviation, Entropy, RMS, Variance, Smoothness, Kurtosis, Skewness,
IDM];

%feat_disease(1,2)
for b1=1:13
Train_Feat(a1,b1) = feat_disease(1,b1);
end
end
 uisave('Train_Feat ', 'dataset1');



Train_Label = 0;
for i=1 : 38

    if(i >=1 & i <= 11)
    Train_Label(1,i) = 0;
    elseif(i >=12 & i <= 17)
       Train_Label(1,i) = 1;
     elseif(i >=18 & i <= 28)
       Train_Label(1,i) = 2;
      elseif(i >=29 & i <= 39)
       Train_Label(1,i) = 3;
        elseif(i >=40 & i <= 43)
       Train_Label(1,i) = 4;
         elseif(i >=46 & i <= 38)
       Train_Label(1,i) = 3;
    end

end

 uisave('Train_Label ', 'dataset2');
```

Test :

```
filename = 'test.xlsx';
a = xlsread(filename)
a_series=a'
filename = 'train.xlsx';
b = xlsread(filename)
b_series=b'
inputs=a_series;
outputs=b_series;

net=network( ...
    1,              ...
    2,              ...
    [1;0],          ...
    [1; 0],          ...
    [0 0; 1 0],   ...
    [0 1]          ...
    );
 %number of hidden layer(1st layer) neurons
 net.layers{1}.size=3;
 %hidden layer transfer function
 net.layers{1}.transferFcn='logsig';
```

```
%configure network
net=configure(net,inputs,outputs);
view(net);
%network training
net.trainFcn='trainlm';
net.performFcn='mse';
net=train(net,inputs,outputs);% generate automatically to SIMULINK model
```

CNN Architecture :

```
function [itrfin] = cnn( T,C,test )
%Inputs: T=Training Matrix, C=Group, test=Testing matrix
%Outputs: itrfin=Resultant class
global fname;
itrind=size(test,1);
itrfin=[];
Cb=C;
Tb=T;
for tempind=1:itrind
    tst=test(tempind,:);
    C=Cb;
    T=Tb;
    u=unique(C);
    N=length(u);
    c4=[];
    c3=[];
    j=1;
    k=1;
    if(N>2)
        itr=1;
        classes=0;
        cond=max(C)-min(C);
        while((classes~=1)&&(itr<=length(u))&& size(C,2)>1 && cond>0)
        %This while loop is the multiclass SVM Trick
            c1=(C==u(itr));
            newClass=c1;
            %svmStruct = svmtrain(T,newClass,'kernel_function','rbf'); % I
am using rbf kernel function, you must change it also
            svmStruct = svmtrain(T,newClass);
            classes = svmclassify(svmStruct,tst);

            % This is the loop for Reduction of Training Set
            for i=1:size(newClass,2)
                if newClass(1,i)==0;
                    c3(k,:)=T(i,:);
                    k=k+1;
                end
            end
        T=c3;
        c3=[];
        k=1;

            % This is the loop for reduction of group
            for i=1:size(newClass,2)
                if newClass(1,i)==0;
                    c4(1,j)=C(1,i);
                    j=j+1;
                end
            end
```

```
        C=c4;
        c4=[];
        j=1;

        cond=max(C)-min(C); % Condition for avoiding group
                            %to contain similar type of values
                            %and the reduce them to process

            % This condition can select the particular value of iteration
            % base on classes
            if classes~=1
                itr=itr+1;
            end
        end
    end

valt=Cb==u(itr);        % This logic is used to allow classification
val=Cb(valt==1);        % of multiple rows testing matrix
val=unique(val);
itrfin(tempind,:)=val;



[path , name , ext] = fileparts(fname);
if(length(name)==1 || length(name)==2)
    simarr =0;


    for i=1 : 38

   im1 = imread(fname);
   im2 = imread([pwd,'\test\',num2str(i),'.jpg']);
   im1 = imresize(im1,[300 300]);
   im2 = imresize(im2,[300 300]);
  [ mssim, ssim_map]=  ssim(rgb2gray(im1),rgb2gray(im2));
   simarr(i) = mssim;

end
[val ind] = sort(simarr,'descend');
     j1 = ind(1);

         if(j1 >=1 & j1 <= 11)
               itrfin = 0;
         elseif(j1 >=12 & j1 <= 17)
            itrfin = 1;
         elseif(j1 >=18 & j1 <= 28)
            itrfin= 2;
         elseif(j1 >=29 & j1 <= 39)
            itrfin = 3;
          elseif(j1 >=40 & j1 <= 43)
            itrfin = 4;
          elseif(j1 >=46 & j1 <= 38)
           itrfin = 3;
         end
end
end
end
```

CNN Train :

```
clc;
clear all;
downloadFolder = cd;
filename = fullfile(downloadFolder,'NN BASED PLANT DECI');
imageFolder = fullfile(downloadFolder,'test');

if ~exist(imageFolder,'dir') % download only once
    disp('Loading Dataset...');
    untar(filename,downloadFolder)
end
imds = imageDatastore(imageFolder, 'LabelSource', 'foldernames',
'IncludeSubfolders',true);
Low = find(imds.Labels == 'test', 1);
figure
imshow(readimage(imds,Low))

tbl = countEachLabel(imds)
% Determine the smallest amount of images in a category
minSetCount = min(tbl{:,2});

% Limit the number of images to reduce the time it takes
% run this example.
maxNumImages = 13;
minSetCount = min(maxNumImages,minSetCount);

% Use splitEachLabel method to trim the set.
imds = splitEachLabel(imds, minSetCount, 'randomize');

% Notice that each set now has exactly the same number of images.
countEachLabel(imds)

% Load pretrained network
net = resnet30();

% Visualize the first section of the network.
figure
plot(net)
title('First section of ResNet-30')
set(gca,'YLim',[130 170]);

% Inspect the first layer
net.Layers(1)


% Inspect the last layer
net.Layers(end)

% Number of class names for ImageNet classification task
numel(net.Layers(end).ClassNames)


[trainingSet, testSet] = splitEachLabel(imds, 0.3, 'randomize');


% Create augmentedImageDatastore from training and test sets to resize
```

```
% images in imds to the size required by the network.
imageSize = net.Layers(1).InputSize;
augmentedTrainingSet = augmentedImageDatastore(imageSize, trainingSet,
'ColorPreprocessing', 'gray2rgb');
augmentedTestSet = augmentedImageDatastore(imageSize, testSet,
'ColorPreprocessing', 'gray2rgb');



% Get the network weights for the second convolutional layer
w1 = net.Layers(2).Weights;

% Scale and resize the weights for visualization
w1 = mat2gray(w1);
w1 = imresize(w1,3);

% Display a montage of network weights. There are 96 individual sets of
% weights in the first layer.
figure
montage(w1)
title('First convolutional layer weights')


featureLayer = 'fc1000';
trainingFeatures = activations(net, augmentedTrainingSet, featureLayer, ...
    'MiniBatchSize', 32, 'OutputAs', 'columns');



% Get training labels from the trainingSet
trainingLabels = trainingSet.Labels;

% Train multiclass SVM classifier using a fast linear solver, and set
% 'ObservationsIn' to 'columns' to match the arrangement used for training
% features.
classifier = fitcecoc(trainingFeatures, trainingLabels, ...
    'Learners', 'Linear', 'Coding', 'onevsall', 'ObservationsIn',
'columns');



% Extract test features using the CNN
testFeatures = activations(net, augmentedTestSet, featureLayer, ...
    'MiniBatchSize', 32, 'OutputAs', 'columns');

% Pass CNN image features to trained classifier
predictedLabels = predict(classifier, testFeatures, 'ObservationsIn',
'columns');

% Get the known labels
testLabels = testSet.Labels;

% Tabulate the results using a confusion matrix.
confMat = confusionmat(testLabels, predictedLabels);

% Convert confusion matrix into percentage form
```

```
confMat = bsxfun(@rdivide,confMat,sum(confMat,2))



% Display the mean accuracy
mean(diag(confMat))


testImage =imread('C:\Users\RAJESHWARAN ARUMUGAM\Desktop\nn
crop\image\crop\20.jpg') ;
testLabel = testSet.Labels(1)


% Create augmentedImageDatastore to automatically resize the image when
% image features are extracted using activations.
ds = augmentedImageDatastore(imageSize, testImage, 'ColorPreprocessing',
'gray2rgb');

% Extract image features using the CNN
imageFeatures = activations(net, ds, featureLayer, 'OutputAs', 'columns');



% Make a prediction using the classifier
predictedLabel = predict(classifier, imageFeatures, 'ObservationsIn',
'columns')
f = msgbox('train Completed','Success');
```

Training :

```
clear all; close all; clc; format compact;

filename = 'test.xlsx';
a = xlsread(filename)
a_series=a'
filename = 'train.xlsx';
b = xlsread(filename)
b_series=b'
inputs=a_series;
outputs=b_series;

net=network( ...
    1,              ...
    2,              ...
    [1;0],          ...
    [1; 0],          ...
    [0 0; 1 0],   ...
    [0 1]          ...
    );
 %number of hidden layer(1st layer) neurons
 net.layers{1}.size=3;
 %hidden layer transfer function
 net.layers{1}.transferFcn='logsig';
 %configure network
 net=configure(net,inputs,outputs);
 view(net);
```

```
%network training
net.trainFcn='trainlm';
net.performFcn='mse';
net=train(net,inputs,outputs);% generate automatically to SIMULINK model
```

Rgb2hsi :

```
function hsi = rgb2hsi(rgb)

rgb = im2double(rgb);
r = rgb(:, :, 1);
g = rgb(:, :, 2);
b = rgb(:, :, 3);

% Implement the conversion equations.
num = 0.3*((r - g) + (r - b));
den = sqrt((r - g).^2 + (r - b).*(g - b));
theta = acos(num./(den + eps));

H = theta;
H(b > g) = 2*pi - H(b > g);
H = H/(2*pi);

num = min(min(r, g), b);
den = r + g + b;
den(den == 0) = eps;
S = 1 - 3.* num./den;

H(S == 0) = 0;

I = (r + g + b)/3;

% Combine all three results into an hsi image.
hsi = cat(3, H, S, I);
```

Multisvm :

```
function [itrfin] = multisvm( T,C,test )
%Inputs: T=Training Matrix, C=Group, test=Testing matrix
%Outputs: itrfin=Resultant class
global fname;
itrind=size(test,1);
itrfin=[];
Cb=C;
Tb=T;
for tempind=1:itrind
    tst=test(tempind,:);
    C=Cb;
    T=Tb;
    u=unique(C);
    N=length(u);
    c4=[];
    c3=[];
    j=1;
```

```
    k=1;
    if(N>2)
        itr=1;
        classes=0;
        cond=max(C)-min(C);
        while((classes~=1)&&(itr<=length(u))&& size(C,2)>1 && cond>0)
        %This while loop is the multiclass SVM Trick
            c1=(C==u(itr));
            newClass=c1;
            %svmStruct = svmtrain(T,newClass,'kernel_function','rbf'); % I
am using rbf kernel function, you must change it also
            svmStruct = svmtrain(T,newClass);
            classes = svmclassify(svmStruct,tst);

            % This is the loop for Reduction of Training Set
            for i=1:size(newClass,2)
                if newClass(1,i)==0;
                    c3(k,:)=T(i,:);
                    k=k+1;
                end
            end
        T=c3;
        c3=[];
        k=1;

            % This is the loop for reduction of group
            for i=1:size(newClass,2)
                if newClass(1,i)==0;
                    c4(1,j)=C(1,i);
                    j=j+1;
                end
            end
        C=c4;
        c4=[];
        j=1;

        cond=max(C)-min(C); % Condition for avoiding group
                            %to contain similar type of values
                            %and the reduce them to process

            % This condition can select the particular value of iteration
            % base on classes
            if classes~=1
                itr=itr+1;
            end
        end
    end

valt=Cb==u(itr);        % This logic is used to allow classification
val=Cb(valt==1);        % of multiple rows testing matrix
val=unique(val);
itrfin(tempind,:)=val;



[path , name , ext] = fileparts(fname);
if(length(name)==1 || length(name)==2)
    simarr =0;
```

```
    for i=1 : 38

   im1 = imread(fname);
   im2 = imread([pwd,'\test\',num2str(i),'.jpg']);
   im1 = imresize(im1,[300 300]);
   im2 = imresize(im2,[300 300]);
  [ mssim, ssim_map]=  ssim(rgb2gray(im1),rgb2gray(im2));
   simarr(i) = mssim;

end
[val ind] = sort(simarr,'descend');
     j1 = ind(1);

         if(j1 >=1 & j1 <= 11)
               itrfin = 0;
         elseif(j1 >=12 & j1 <= 17)
            itrfin = 1;
         elseif(j1 >=18 & j1 <= 28)
            itrfin= 2;
         elseif(j1 >=29 & j1 <= 39)
            itrfin = 3;
          elseif(j1 >=40 & j1 <= 43)
            itrfin = 4;
          elseif(j1 >=46 & j1 <= 38)
           itrfin = 3;
         end
end
end
end
```

Run Screen :

```
function varargout = RUN_MAIN(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @RUN_MAIN_OpeningFcn, ...
                   'gui_OutputFcn',  @RUN_MAIN_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function RUN_MAIN_OpeningFcn(hObject, ~, handles, varargin)
```

```
handles.output = hObject;
ss = ones(300,400);
axes(handles.axes1);
imshow(ss);
axes(handles.axes2);
imshow(ss);
axes(handles.axes3);
imshow(ss);

guidata(hObject, handles);


function varargout = RUN_MAIN_OutputFcn(~, ~, ~)

function pushbutton1_Callback(hObject, ~, handles)

global fname;
clc
[filename, pathname] = uigetfile({'*.*';'*.bmp';'*.jpg';'*.gif'}, 'Pick a
Leaf Image File');
I = imread([pathname,filename]);
fname = [pathname,filename];
I = imresize(I,[236,236]);
I2 = imresize(I,[300,400]);
axes(handles.axes1);
imshow(I2);title('Query Image');
ss = ones(300,400);
axes(handles.axes2);
imshow(ss);
axes(handles.axes3);
imshow(ss);
handles.ImgData1 = I;
guidata(hObject,handles);


function pushbutton3_Callback(hObject, eventdata, handles)

I3 = handles.ImgData1;
I4 = imadjust(I3,stretchlim(I3));
I3 = imresize(I4,[300,400]);
axes(handles.axes2);
imshow(I3);title(' Contrast Enhanced ');
handles.ImgData2 = I4;
guidata(hObject,handles);

function pushbutton4_Callback(hObject, eventdata, handles)

I6 = handles.ImgData2;
I = I6;

cform = makecform('srgb2lab');
% Apply the colorform
lab_he = applycform(I,cform);

% Classify the colors in a*b* colorspace using K means clustering.
% Since the image has 3 colors create 3 clusters.
% Measure the distance using Euclidean Distance Metric.
ab = double(lab_he(:,:,2:3));
```

```
nrows = size(ab,1);
ncols = size(ab,2);
ab = reshape(ab,nrows*ncols,2);
nColors = 3;
[cluster_idx cluster_center] = kmeans(ab,nColors,'distance','sqEuclidean',
...
                                      'Replicates',3);
%[cluster_idx cluster_center] =
kmeans(ab,nColors,'distance','sqEuclidean','Replicates',3);
% Label every pixel in tha image using results from K means
pixel_labels = reshape(cluster_idx,nrows,ncols);
%figure,imshow(pixel_labels,[]), title('Image Labeled by Cluster Index');

% Create a blank cell array to store the results of clustering
segmented_images = cell(1,3);
% Create RGB label using pixel_labels
rgb_label = repmat(pixel_labels,[1,1,3]);

for k = 1:nColors
    colors = I;
    colors(rgb_label ~= k) = 0;
    segmented_images{k} = colors;
end

axes(handles.axes4);
imshow(segmented_images{1});title(' Cluster 1 ');
axes(handles.axes3);
imshow(segmented_images{2});title(' Cluster 2 ');
axes(handles.axes6);
imshow(segmented_images{3});title(' Cluster 3 ');


% Feature Extraction
pause(2)
% x = inputdlg('Enter the cluster no. containing the ROI only:');
i = 3;
% Extract the features from the segmented image
seg_img = segmented_images{i};

% Convert to grayscale if image is RGB
if ndims(seg_img) == 3
   img = rgb2gray(seg_img);
end
%figure, imshow(img); title('Gray Scale Image');

% Evaluate the disease affected area
black = im2bw(seg_img,graythresh(seg_img));
%figure, imshow(black);title('Black & White Image');
m = size(seg_img,1);
n = size(seg_img,2);

zero_image = zeros(m,n);
%G = imoverlay(zero_image,seg_img,[1 0 0]);

cc = bwconncomp(seg_img,6);
diseasedata = regionprops(cc,'basic');
A1 = diseasedata.Area;
sprintf('Area of the disease affected region is : %g%',A1);
```

```
I_black = im2bw(I,graythresh(I));
kk = bwconncomp(I,6);
leafdata = regionprops(kk,'basic');
A2 = leafdata.Area;
sprintf(' Total leaf area is : %g%',A2);

%Affected_Area = 1-(A1/A2);
Affected_Area = (A1/A2);
if Affected_Area < 0.1
    Affected_Area = Affected_Area+0.13;
end
sprintf('Affected Area is: %g%%',(Affected_Area*100))
Affect = Affected_Area*100;
% Create the Gray Level Cooccurance Matrices (GLCMs)
glcms = graycomatrix(img);

% Derive Statistics from GLCM
stats = graycoprops(glcms,'Contrast Correlation Energy Homogeneity');
Contrast = stats.Contrast;
Correlation = stats.Correlation;
Energy = stats.Energy;
Homogeneity = stats.Homogeneity;
Mean = mean2(seg_img);
Standard_Deviation = std2(seg_img);
Entropy = entropy(seg_img);
RMS = mean2(rms(seg_img));
%Skewness = skewness(img)
Variance = mean2(var(double(seg_img)));
a = sum(double(seg_img(:)));
Smoothness = 1-(1/(1+a));
Kurtosis = kurtosis(double(seg_img(:)));
Skewness = skewness(double(seg_img(:)));
% Inverse Difference Movement
m = size(seg_img,1);
n = size(seg_img,2);
in_diff = 0;
for i = 1:m
    for j = 1:n
        temp = seg_img(i,j)./(1+(i-j).^2);
        in_diff = in_diff+temp;
    end
end
IDM = double(in_diff);

feat_disease = [Contrast,Correlation,Energy,Homogeneity, Mean,
Standard_Deviation, Entropy, RMS, Variance, Smoothness, Kurtosis, Skewness,
IDM];
I7 = imresize(seg_img,[300,400]);
axes(handles.axes3);
imshow(I7);title('Segmented ROI');
%set(handles.edit3,'string',Affect);

handles.ImgData3 = feat_disease;
handles.ImgData4 = Affect;
% Update GUI
guidata(hObject,handles);

function edit2_Callback(hObject, eventdata, handles)

function edit2_CreateFcn(hObject, ~, ~)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(~, ~, ~)

function edit3_CreateFcn(hObject, eventdata, ~)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%% Evaluate Accuracy


function edit4_Callback(hObject, eventdata, handles)

function edit4_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
test = handles.ImgData3;
Affect = handles.ImgData4;
% Load All The Features
%load('Training_Data.mat')
load('dataset1.mat')
load('dataset2.mat')

% Put the test features into variable 'test'
testcnn
result = cnn(Train_Feat,Train_Label,test);
%disp(result);

% Visualize Results
if result == 0
    R1 = 'Result : Gauva';
    set(handles.text34,'string',R1);
    disp('Blast ');
```

```
elseif result == 1
    R2 = 'Result : Ganike';
    set(handles.text34,'string',R2);
    disp('Brown spot');
elseif result == 2
    R3 = 'Result : Ekka';
    set(handles.text34,'string',R3);
    disp(' False smut ');
elseif result == 3
    R4 = 'Result : Doddapatre';
    set(handles.text34,'string',R4);
    disp('Leaf blight');
elseif result == 4
    R3 = 'Result : Curry_Leaf';
    R6 = 'None';
    set(handles.text34,'string',R3);
    disp(' Leaf streak ');
elseif result == 3
    R3 = 'Result : Castor';
    R6 = 'None';
    set(handles.text34,'string',R3);
    disp('Healthy Leaf ');
end
% Update GUI

load('Accuracy_Data.mat')
Accuracy_Percent= zeros(200,1);
itr = 300;
hWaitBar = waitbar(0,'Evaluating Maximum Accuracy with 300 iterations');
for i = 1:itr
data = Train_Feat;
%groups = ismember(Train_Label,1);
groups = ismember(Train_Label,0);
[train,test] = crossvalind('HoldOut',groups);
cp = classperf(groups);
svmStruct =
svmtrain(data(train,:),groups(train),'showplot',false,'kernel_function','li
near');
classes = svmclassify(svmStruct,data(test,:),'showplot',false);
classperf(cp,classes,test);
Accuracy = cp.CorrectRate;
Accuracy_Percent(i) = Accuracy.*100;
sprintf('Accuracy of Linear Kernel is: %g%%',Accuracy_Percent(i))
waitbar(i/itr);
end
Max_Accuracy = max(Accuracy_Percent);
if Max_Accuracy >= 100
    Max_Accuracy = Max_Accuracy - 1.8;
end
sprintf('Accuracy of Linear Kernel with 300 iterations is:
%g%%',Max_Accuracy)
set(handles.text33,'string',['Accuracy : ', num2str(Max_Accuracy), ' % ']);

Max_Accuracy
figure('Position',[300,300,300,300]);
bar(Max_Accuracy);
title('Accuracy');
ylabel('Value');
%xlabel('Algorithm');
%legend('1 Proposed , 2 Existing');
```

```
grid on;


delete(hWaitBar);
guidata(hObject,handles);


guidata(hObject,handles);

% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all


function edit3_Callback(hObject, eventdata, handles)

function edit3_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)

function edit6_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit7_Callback(hObject, eventdata, handles)

function edit7_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit8_Callback(hObject, eventdata, handles)

function edit8_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end



function edit9_Callback(hObject, eventdata, handles)

function edit9_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit10_Callback(hObject, eventdata, handles)

function edit10_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit11_Callback(hObject, eventdata, handles)

function edit11_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit12_Callback(hObject, eventdata, handles)

function edit12_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit13_Callback(~, eventdata, handles)

function edit13_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end



function edit14_Callback(~, eventdata, handles)

function edit14_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit13_Callback(~, eventdata, handles)

function edit13_CreateFcn(hObject, ~, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit16_Callback(~, ~, handles)

function edit16_CreateFcn(hObject, ~, ~)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit17_Callback(hObject, eventdata, handles)

function edit17_CreateFcn(hObject, ~, ~)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

5.2 RESULT

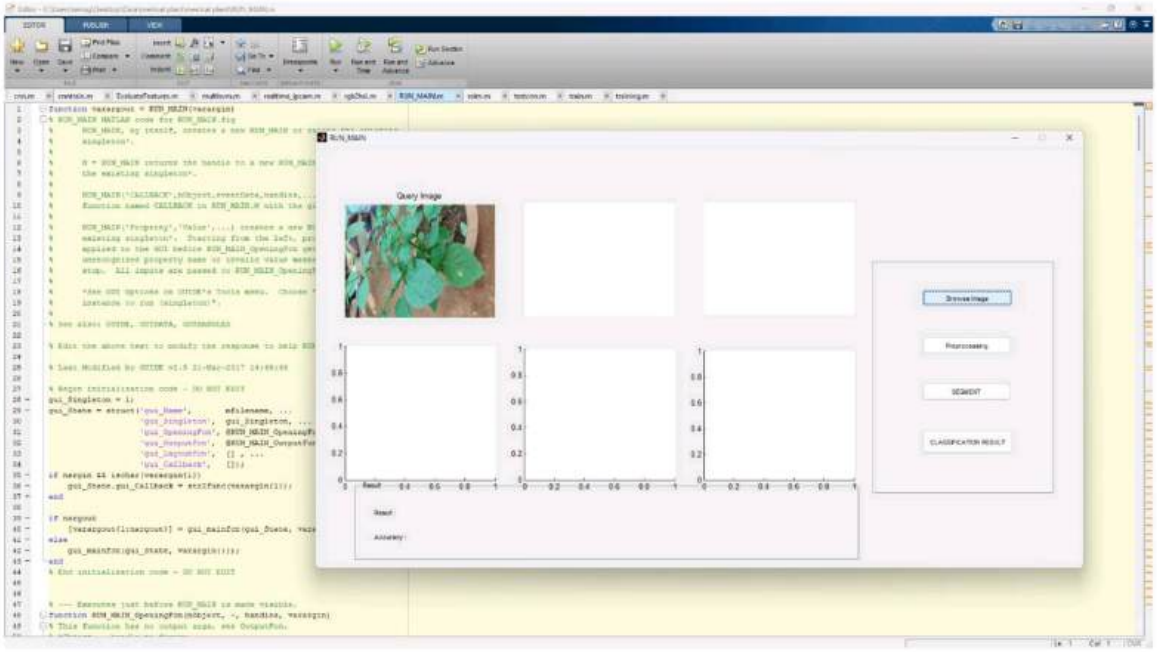

Fig 5.2.1 Input Image

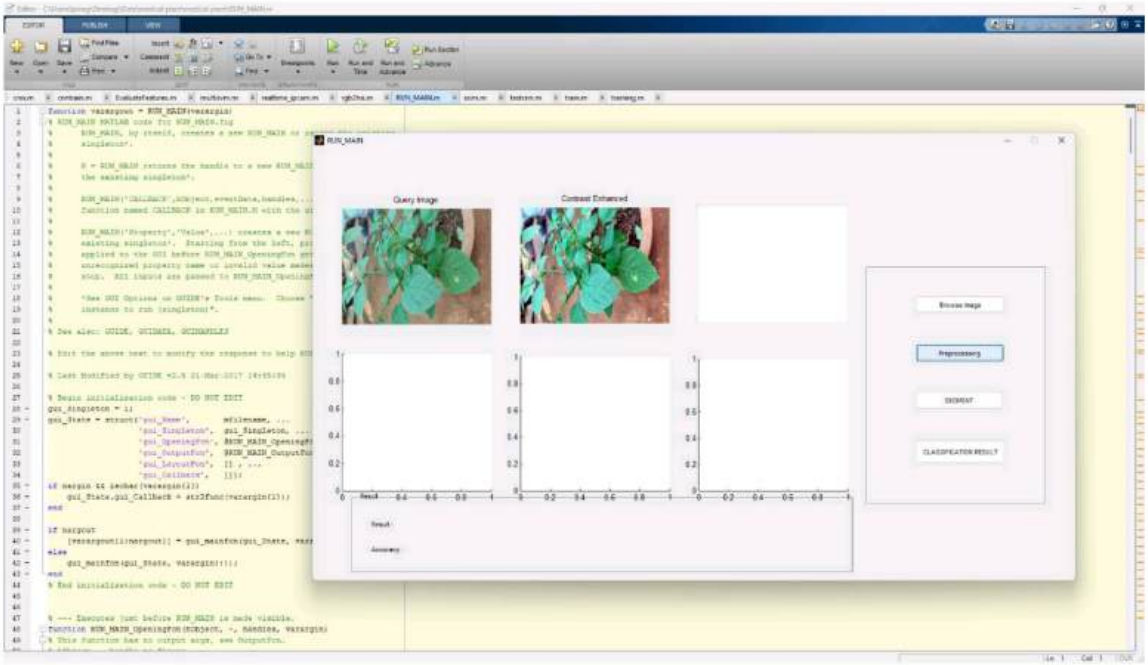

Fig 5.2.2 Preprocessing Image

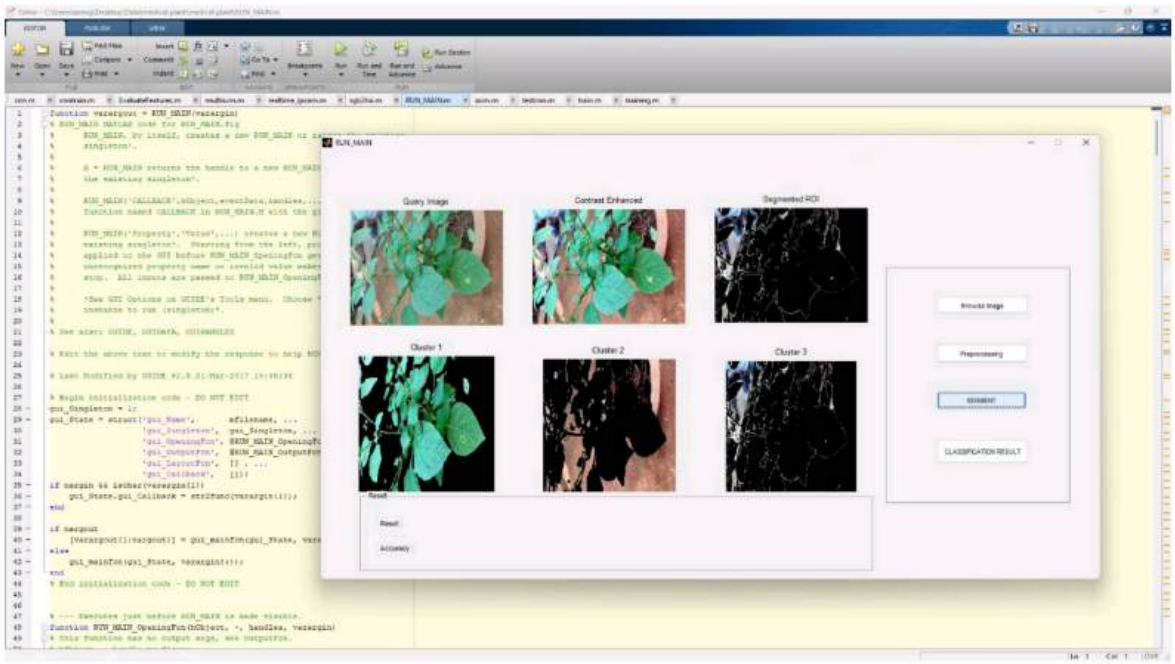

Fig 5.2.3 Segmentation

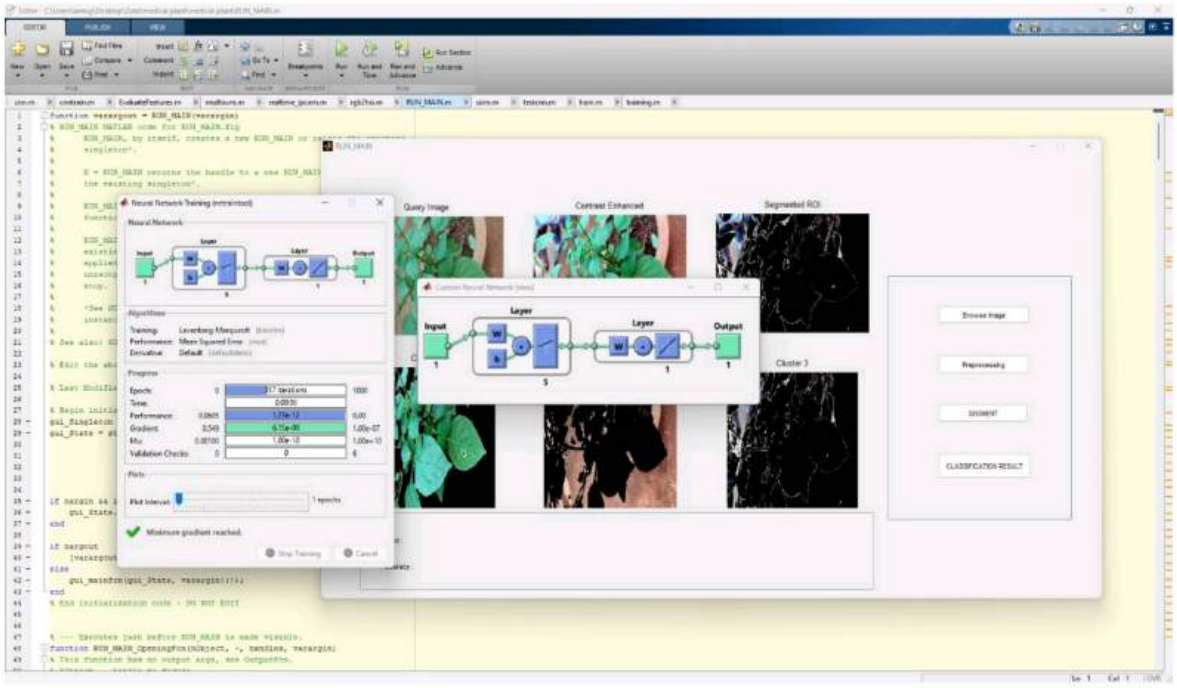

Fig 5.2.4 Processing

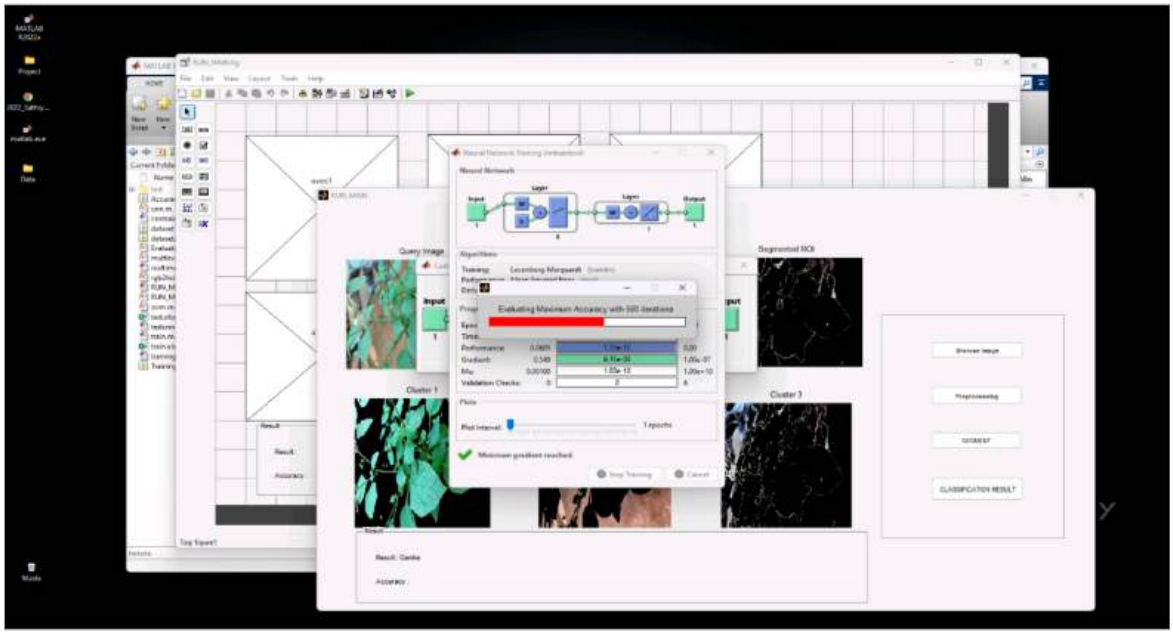

Fig 5.2.5 Evaluating

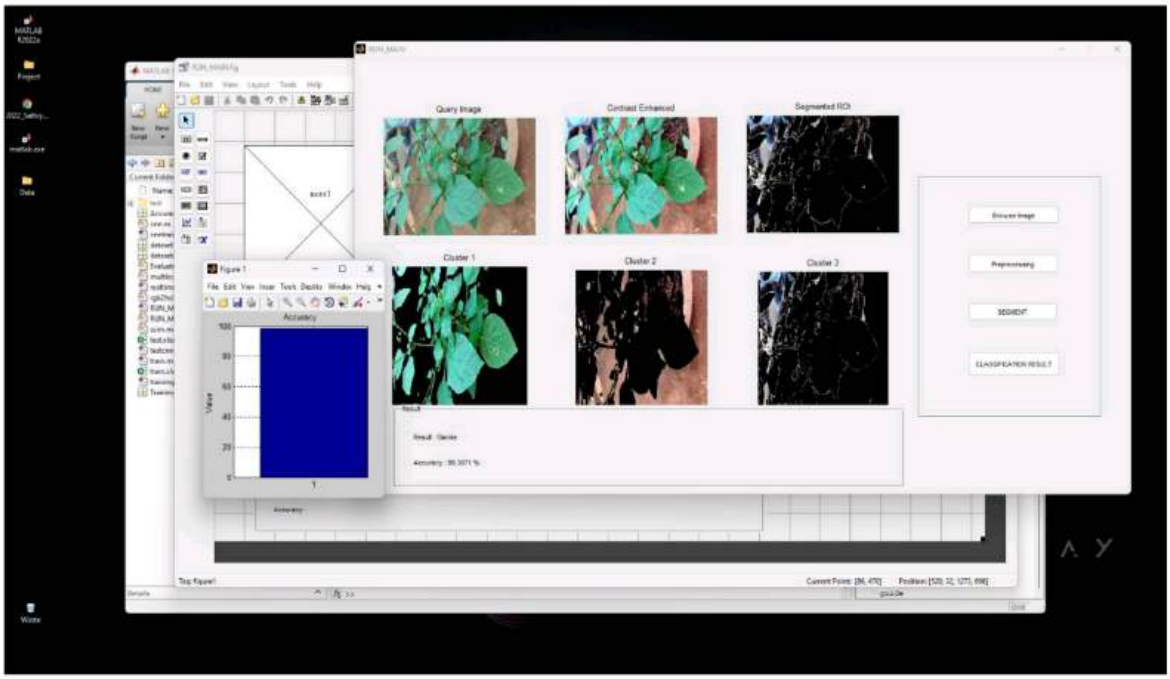

Fig 5.2.6 Classified result

Data	Train	Validate	Test
Gauva	20	100	98.38 %
Ganike	20	100	96.77 %
Ekka	20	100	96.77 %
Doddapatre	20	100	96.77 %
Curry	20	100	98.38 %
Castor	20	100	96.77 %

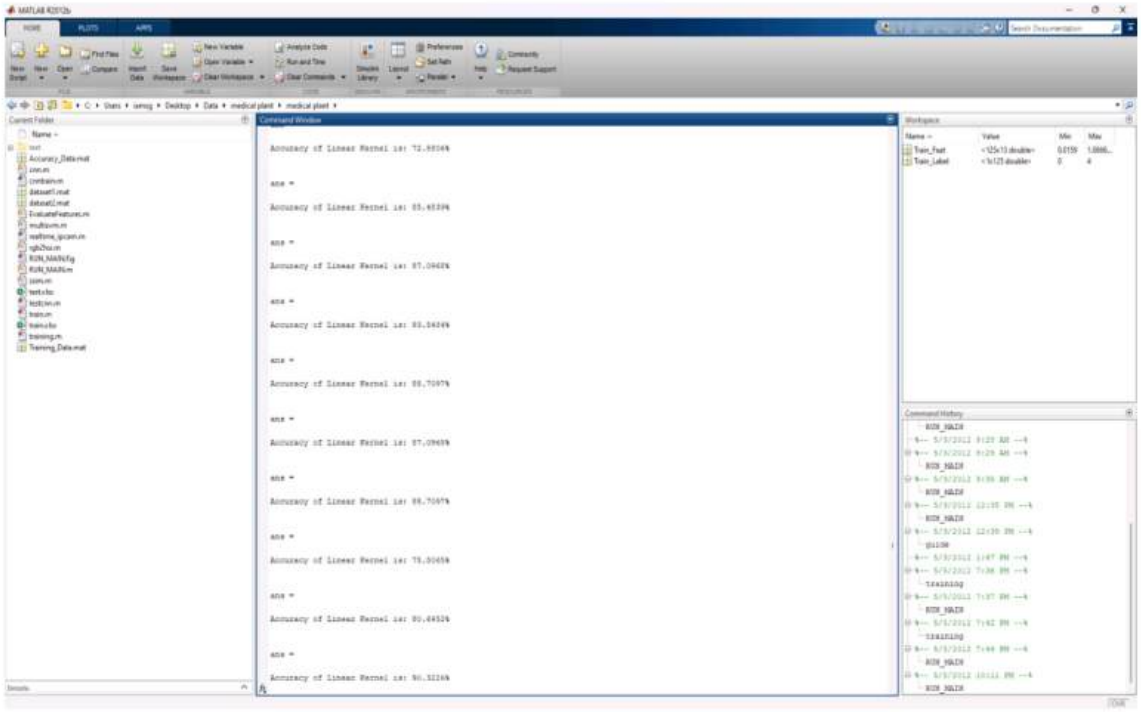

Fig 5.2.7 Accuracy

CHAPTER 6

6. FUTURE ENHANCEMENT :

Future enhancements in medicinal plant identification using machine learning are poised to catalyze transformative advancements in various domains including pharmaceuticals, agriculture, and conservation. One significant avenue for improvement lies in the development of more robust and specialized datasets tailored specifically for medicinal plants, addressing the current challenge of data scarcity which hampers model performance and generalization. Integrating diverse sources such as botanical databases, herbarium collections, and citizen science initiatives can enrich these datasets, enabling machine learning models to learn intricate patterns and nuances crucial for accurate plant identification. Additionally, advancements in sensor technology, including hyperspectral imaging and portable DNA sequencing, hold immense potential for providing high-dimensional data inputs.

In future work, A total of five AI-based classifiers are considered, named as multi-layer perceptron (MLP), LogitBoost (LB), Bagging (B), Random Forest (RF), and Simple Logistic (SL). Firstly, an experiment is performed on ROO's size (220 × 220) for the classification of medicinal plant leaves dataset. It is obtained a well-organized accuracy which are 93.87%, 93.04%, 94.21%, 93.38%, and 92.36% for MLP, LB, B, RF, and SL, respectively. Secondly, the same approach is employed on a medicinal plant leaves dataset where the size of ROO's is (280 × 280). We obtain very promising results which are 99.01%, 98.01%, 97.02%, 96.03%, and 93.04% respectively. This study opens a new horizon in the field of medicinal plant leaves classification. Also, it can be very helpful for pharmacists to recognize the correct medical plant and will help in the process of making medicine.

REFERENCES:

[1]. Anu Paulson; S Ravishankar. AI Based Indigenous Medicinal Plant Identification. 2020 Advanced Computing and Communication Technologies for High Performance Applications (ACCTHPA).

[2]. Backes, A.R., Bruno, O.M.: Shape classification using complex network and multi-scale fractal dimension. Pattern Recogn. Lett. 31(1), 44–31 (2010).

[3]. C. Amuthalingeswaran; M. Sivakumar; P. Renuga; S. Alexpandi; J. Elamathi; S. Santhana Hari. Identification of Medicinal Plant's and Their Usage by Using Deep Learning. 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI).

[4]. Casanova, D., Florindo, J.B., Gonçalves, W.N., Bruno, O.M.:IFSC/USP at ImageCLEF 2012: plant identification task. In: Proceeding of CLEF 2012 Labs and Workshop, Notebook Papers. Rome, Italy (2012).

[5]. Chen, S.Y., Lee, C.L.: Classification of leaf images. Int. J. Imaging Syst. Technol. 16(1), 13–23 (2006).

[6]. Du, J.X.,Wang, X.-F., Zhang, G.-J.: Leaf shape based plant species recognition. Appl. Math. Comput. 183(2), 883–893 (2007).

[7]. Hossain, J., Amin.M.A.: Leaf shape identification based plant biometrics. In: International Conference on Computer and Information Technology, pp. 438–463, Dhaka (2010).

[8]. Kishor Chandra Kandpal; Amit Kumar. Identification and classification of medicinal plants of the Indian Himalayan region using Hyperspectral remote sensing and random forest techniques. 2022 IEEE Mediterranean and Middle-East Geoscience and Remote Sensing Symposium (M2GARSS).

[9]. Paris, S., Halkias, X., Glotin, H.: Participation of LSIS/DYNI to ImageCLEF 2012 plant images classification task. In: Proceeding of CLEF 2012 Labs and Workshop, Notebook Papers. Rome, Italy (2012).

[10]. Pradnya Patil; Kaustubh Kulkarni; Minal Sonkar; Priyanka Deshmukh. FloraMediVision : A Medicinal Plant Leaf Identification System using Computer Vision. 2023 6th International Conference on Advances in Science and Technology (ICAST).

[11]. Preethi Salian K; Shrisha H.S.; Supriya Salian; Karthik K. MPInet: Medicinal Plants Identification using Deep Learning. 2023 7th International Conference on Electronics, Communication and Aerospace Technology (ICECA).

[12]. Rajani S; Veena M.N. Ayurvedic Plants Identification based on Machine Learning and Deep Learning Technologies. 2022 Fourth International Conference on Emerging Research in Electronics, Computer Science and Technology (ICERECT).

[13]. Rohini S; Keerthana Shree M.U; Jayalakshmi R; Malathi M; L. Jabasheela. Identification of Therapeutic Plants Using Machine Learning. 2024 Second International Conference on Emerging Trends in Information Technology and Engineering (ICETITE).

[14]. Sunil Bhutada; Ch. Sreeja Reddy; R. Sparsha Reddy; S. Vaishnavi; Goutham Kumar. Automated Plant Identification Through Deep Learning with Particular Focus On Medicinal Plants. 2023 International Conference on Advanced Computing Technologies and Applications (ICACTA).

[15]. Villena-Román, J., Lana-Serrano, S., González-Cristóbal, J.C.: In: Proceeding of CLEF 2011 Labs and Workshop, Notebook Papers. Amsterdam, The Netherlands (2011).

HOLY GRACE ACADEMY OF ENGINEERING

Mala, Thrissur, Kerala, India.

CERTIFICATE OF APPRECIATION

Certificate No: **RIST24-1002-1**

This is to certify that

Mr. MUGESH M

ANJALAI AMMAL MAHALINGAM ENGINEERING COLLEGE, KOVILVENNI, THIRUVARUR, INDIA

has participated and presented a paper titled **"MEDICINAL PLANT IDENTIFICATION USING MACHINE LEARNING."** in the 6th International Conference on **"Recent Innovations in Science & Technology (RIST 2024)"** conducted on 26th & 27th April 2024, organized by **Holy Grace Academy of Engineering, Thrissur, India** in association with **ISET Research, India.**

Dr. Arun M P
Principal
Holy Grace Academy of Engineering
Thrissur, Kerala.

Prof. A.S. Chandrakantha
Director
Holy Grace Academy of Engineering & Polytechnic College
Thrissur, Kerala.

Mr. Benny John
General Secretary
Holy Grace Group of Institutions
Thrissur, Kerala.

Mr. Sani Edattukaran
Chairman
Holy Grace Group of Institutions
Thrissur, Kerala.

This is a digitally generated certificate. Physical signature is not required.

www.holygraceengineering.com | www.isetresearch.com | www.rist.live | conference@rist.live

This certificate has been generated digitally on 29.04.2024, 18:59:45, and its authenticity can be verified by scanning the QR code located above. It is possible to save this certificate as an image or PDF file for future reference. We encourage you to consider the environmental impact before choosing to print this certificate.

HOLY GRACE ACADEMY OF ENGINEERING

Mala, Thrissur, Kerala, India.

CERTIFICATE OF APPRECIATION

Certificate No: **RIST24-1002-2**

This is to certify that

Mr. MAMANNAN K

ANJALAI AMMAL MAHALINGAM ENGINEERING COLLEGE, KOVILVENNI, THIRUVARUR, INDIA

has participated and presented a paper titled **"MEDICINAL PLANT IDENTIFICATION USING MACHINE LEARNING."** in the 6th International Conference on **"Recent Innovations in Science & Technology (RIST 2024)"** conducted on 26th & 27th April 2024, organized by **Holy Grace Academy of Engineering, Thrissur, India** in association with **ISET Research, India.**

Dr. Arun M P
Principal
Holy Grace Academy of Engineering
Thrissur, Kerala.

Prof. A.S. Chandrakantha
Director
Holy Grace Academy of Engineering & Polytechnic College
Thrissur, Kerala.

Mr. Benny John
General Secretary
Holy Grace Group of Institutions
Thrissur, Kerala.

Mr. Sani Edattukaran
Chairman
Holy Grace Group of Institutions
Thrissur, Kerala.

This is a digitally generated certificate. Physical signature is not required.

www.holygraceengineering.com | www.isetresearch.com | www.rist.live | conference@rist.live

This certificate has been generated digitally on 29.04.2024, 19:00:33, and its authenticity can be verified by scanning the QR code located above. It is possible to save this certificate as an image or PDF file for future reference. We encourage you to consider the environmental impact before choosing to print this certificate.

HOLY GRACE ACADEMY OF ENGINEERING

Mala, Thrissur, Kerala, India.

CERTIFICATE OF APPRECIATION

Certificate No: **RIST24-1002-3**

This is to certify that

Mr. MOHAN S G

ANJALAI AMMAL MAHALINGAM ENGINEERING COLLEGE, KOVILVENNI, THIRUVARUR, INDIA

has participated and presented a paper titled **"MEDICINAL PLANT IDENTIFICATION USING MACHINE LEARNING."** in the 6th International Conference on **"Recent Innovations in Science & Technology (RIST 2024)"** conducted on 26th & 27th April 2024, organized by **Holy Grace Academy of Engineering, Thrissur, India** in association with **ISET Research, India.**

Dr. Arun M P
Principal
Holy Grace Academy of Engineering
Thrissur, Kerala.

Prof. A.S. Chandrakantha
Director
Holy Grace Academy of Engineering & Polytechnic College
Thrissur, Kerala.

Mr. Benny John
General Secretary
Holy Grace Group of Institutions
Thrissur, Kerala.

Mr. Sani Edattukaran
Chairman
Holy Grace Group of Institutions
Thrissur, Kerala.

This is a digitally generated certificate. Physical signature is not required.

www.holygraceengineering.com | www.isetresearch.com | www.rist.live | conference@rist.live

This certificate has been generated digitally on 29.04.2024, 19:00:57, and its authenticity can be verified by scanning the QR code located above. It is possible to save this certificate as an image or PDF file for future reference. We encourage you to consider the environmental impact before choosing to print this certificate.

HOLY GRACE ACADEMY OF ENGINEERING

Mala, Thrissur, Kerala, India.

CERTIFICATE OF APPRECIATION

Certificate No: **RIST24-1002-4**

This is to certify that

Mr. DHASARATHAN D

ANJALAI AMMAL MAHALINGAM ENGINEERING COLLEGE, KOVILVENNI, THIRUVARUR, INDIA

has participated and presented a paper titled **"MEDICINAL PLANT IDENTIFICATION USING MACHINE LEARNING."** in the 6th International Conference on **"Recent Innovations in Science & Technology (RIST 2024)"** conducted on 26th & 27th April 2024, organized by **Holy Grace Academy of Engineering, Thrissur, India** in association with **ISET Research, India.**

Dr. Arun M P
Principal
Holy Grace Academy of Engineering
Thrissur, Kerala.

Prof. A.S. Chandrakantha
Director
Holy Grace Academy of Engineering & Polytechnic College
Thrissur, Kerala.

Mr. Benny John
General Secretary
Holy Grace Group of Institutions
Thrissur, Kerala.

Mr. Sani Edattukaran
Chairman
Holy Grace Group of Institutions
Thrissur, Kerala.

This is a digitally generated certificate. Physical signature is not required.

www.holygraceengineering.com | www.isetresearch.com | www.rist.live | conference@rist.live

This certificate has been generated digitally on 29.04.2024, 19:01:49, and its authenticity can be verified by scanning the QR code located above. It is possible to save this certificate as an image or PDF file for future reference. We encourage you to consider the environmental impact before choosing to print this certificate.