Chapter No. 1

INTRODUCTION

1.1 Introduction to Project

India is an agricultural country, where most of the population depends on agricultural products. So, the cultivation can be improved by technological support. Diseases may cause by pathogen in plant at any environmental condition. In most of the cases diseases are seen on the leaves of the plants, so the detection of disease plays an important role in successful cultivation of crops. There are lots of techniques to detect the different types of diseases in plants in its early stages. A conventional method of plant disease detection in naked eye observation methods is non effective for large crops. Health monitoring and disease detection on plant is very critical for sustainable agriculture. It is very difficult to monitor the plant diseases manually. It requires tremendous amount of work, expertise in the plant diseases, and also require the excessive processing time.

Using digital image processing and machine learning, the disease detection in plant is efficient, less time consuming and accurate. This technique saves time, efforts, labours and use of pesticides. Disease detection involves the steps like image acquisition, image pre-processing, image segmentation, feature extraction and classification. Developments in image processing methods and machine learning techniques have lead many automatic detection algorithms. The main aim of machine learning is to understand the training data and fit that training data into models that should be useful to the people. For this approach automatic classifier Convolutional Neural Networks (CNN) model is used for classification based on learning with some training samples. This can assist in good decisions making and predicting the correct output using the large amount of training data.

1.2 Purpose of the Project

A need of this project is the ease of detection of the plant disease by the farmers in a very simple manner as farmers cannot predict the disease just by looking at the leaf.

The main purpose is to make an efficient use of Machine Learning Algorithms which reduces time and cost for farmer to detect the plant disease by using feature extraction methods where features such as shape, colour, and texture are taken into consideration.

1.3 Problem definition

1.3.1 Existing System

The existing method for plant disease detection is simply naked eye observation by experts through which identification and detection of plant diseases is done. For doing so, a large team of experts as well as continuous monitoring of plant is required, which costs very high when we do with large farms. At the same time, in some countries, farmers do not have proper facilities or even idea that they can contact to experts. Due to which consulting experts even cost high as well as time consuming too. Plant disease identification by visual way is more laborious task and at the same time, less accurate and can be done only in limited areas. Whereas if automatic detection technique is used it will take less efforts, less time and become more accurate.

1.3.2 Proposed System

The main purpose of proposed system is to detect the diseases of plant leaves by using feature extraction methods where features such as shape, colour, and texture are taken into consideration. Convolutional neural network (CNN), a machine learning technique is used in classifying the plant leaves into healthy or diseased and if it is a diseased plant leaf, CNN will give the name of that particular disease. Suggesting remedies for particular disease is made which will help in growing healthy plants and improve the productivity.

First the images of various leaves are acquired using high resolution camera so as to get the better results & efficiency. Then image processing techniques are applied to these images to extract useful features which will be required for further analysis. The basic steps of the system are summarized as:

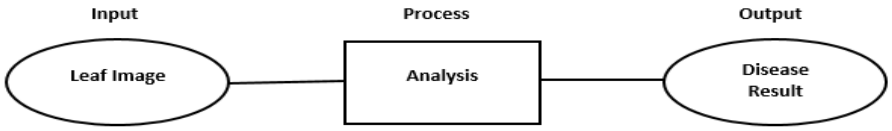

Fig 1.3.2.1 – *Proposed System*

The advantages of proposed algorithm are as follows:

- Use of estimators for automatic Initialization of cluster centers so there is no need of user input at the time of segmentation.
- The detection accuracy is enhanced with proposed algorithm.

- Proposed method is fully automatic while existing methods require user input to select the best segmentation of input image.
- It also provides environment friendly recovery measures of the identified disease.

1.4 Scope of the Project

This project will be very helpful to farmers in rural areas and also will help them in saving their yields from diseases as farmers lose a huge amount of their cultivated crops because of diseases and this system will help them to avoid the similar situation. Early and accurate detection of diseases in the initial phases will help to reduce the loss and amount of damage caused to the leaves. Pre-Detection of diseases will help reduce the cost as well as quality of leaf and so the crops. This can even be applied to large data stored in agricultural department thus leading to better results and faster finding.

1.5 Report organization

The project report is organized as follows: -

Chapter 1: INTRODUCTION

This section focuses on the purpose and scope of our project titled 'Detection of Floral Leaf Affliction using Image Processing Techniques'.

Chapter 2: LITERATURE SURVEY

This section describes the review of research papers in this field, concepts and technologies used to develop the project.

Chapter 3: DESIGN

This section includes the Project Design and UML Diagrams which describes the data and control flow of our project.

Chapter 4: IMPLEMENTATION AND RESULT ANALYSIS

This section deals with the implementation of the various modules of the project.

Chapter 5: CONCLUSION AND FUTURE SCOPE

This section deals with the conclusion and the future scope of the project.

Chapter No. 2

LITERATURE SURVEY

2.1 Introduction

Identification of the plant diseases is the key to preventing the losses in the yield and quantity of the agricultural product. The studies of the plant diseases mean the studies of visually observable patterns seen on the plant. Health monitoring and disease detection on plant is very critical for sustainable agriculture. It is very difficult to monitor the plant diseases manually. It requires tremendous amount of work, expertize in the plant diseases, and also require the excessive processing time. Hence, image processing and Machine learning techniques are used for the detection of plant diseases. Disease detection involves the steps like image acquisition, image pre-processing, image segmentation, feature extraction and classification.

2.1.1 Artificial neural networks

ANNs are computer systems that are modeled after the biological neural networks that make up animal brains. Artificial neurons are a set of connected units or nodes in an ANN that loosely replicate the neurons in a biological brain. Each link can send a signal to other neurons, just like synapses in a human brain.

An artificial neuron receives a signal, analyses it, and then sends signals to neurons it is connected to. Each neuron's output is generated by some non-linear function of the sum of its inputs, and the "signal" at a connection is a real number. Edges is the terms for the connections. The weight of neurons and edges is frequently adjusted as learning progresses. The signal strength at a connection is increased or decreased by the weight.

Neurons may have a threshold that allows them to send a signal only if the aggregate signal exceeds it. Neurons are usually grouped into layers. On their inputs, separate layers may apply different transformations. Signals go from the first layer (input layer) to the last layer (output layer), maybe many times.

2.1.2 Naive Bayes classifiers

NBCs are a type of "probabilistic classifier" based on Bayes' theorem and strong (nave) independence assumptions between features. They are one of the most basic Bayesian network

models, but when combined with kernel density estimation, they can achieve higher levels of accuracy.

The number of parameters required by Nave Bayes classifiers is linear in the number of variables in a learning problem. Maximum-likelihood training can be done in linear time by evaluating a closed-form expression, rather than the time-consuming iterative approximation required by many other forms of classifiers.

Naive Bayes is a straightforward method for building classifiers, which are models that give class labels to problem cases represented as vectors of feature values, with the class labels selected from a limited set.

For training such classifiers, there is no one algorithm, but rather a variety of algorithms based on the same principle: all naive Bayes classifiers assume that the value of one feature is independent of the value of any other feature, given the class variable.

2.1.3 Convolutional Neural Network

CNN is a type of artificial neural network used to interpret visual imagery in deep learning. Shift invariant or SIANN are built on the shared-weight architecture of convolution kernels or filters that slide along input features and produce translation equivariant outputs known as feature maps.

Contrary to popular belief, most convolutional neural networks are merely equivariant to translation, rather than invariant. They're used in image and video recognition, recommender systems, image classification, image segmentation, medical image analysis, natural language processing, brain-computer interfaces, and financial time series. Multilayer perceptron are regularized variants of CNNs.

Multilayer perceptron is typically completely connected networks, meaning that each neuron in one layer is linked to all neurons in the following layer. These networks' "complete connectedness" makes them vulnerable to data over fitting. Regularization, or preventing over fitting, can be accomplished in a variety of methods, including punishing parameters during training (such as weight loss) or reducing connectivity (skipped connections, dropout, etc.)

CNNs take a different approach to regularization: they take advantage of the hierarchical pattern in data and use smaller and simpler patterns imprinted in their filters to assemble patterns of

increasing complexity. As a result, CNNs are at the lower end of the connectivity and complexity spectrum.

2.1.4 Python

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including metaprogramming and metaobjects [magic methods]). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It uses dynamic name resolution (late binding), which binds method and variable names during program execution. Its design offers some support for functional programming in the Lisp tradition. It has filter,mapandreduce functions; list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

Python's developers strive to avoid premature optimization and reject patches to non-critical parts of the CPython reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C; or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

A common neologism in the Python community is pythonic, which has a wide range of meanings related to program style. "Pythonic" code may use Python idioms well, be natural or show fluency in the language, or conform with Python's minimalist philosophy and emphasis on readability. Code that is difficult to understand or reads like a rough transcription from another programming language is called unpythonic. Python users and admirers, especially those considered knowledgeable or experienced, are often referred to as Pythonistas.

2.1.5 Confusion Matrix

A confusion matrix, also known as an error matrix, is a **summarized table** used to assess the performance of a classification model. The number of correct and incorrect predictions are summarized with count values and broken down by each class.

Below is an image of the structure of a 2x2 confusion matrix. To give an example, let's say that there were ten instances where a classification model predicted 'Yes' in which the actual value was 'Yes'. Then the number ten would go in the top left corner in the True Positive quadrant. This leads us to some key terms:

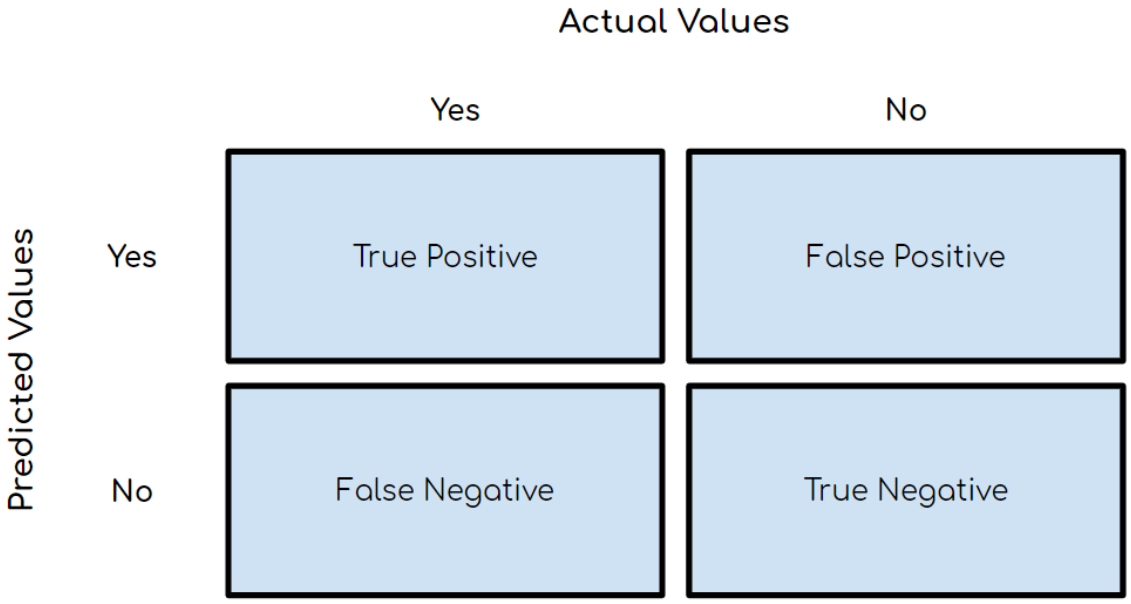

Fig 2.1 Example of Confusion Matrix

Structure for a 2x2 Confusion Matrix

- **Positive (P)**: Observation is positive (eg. **is** a dog).
- **Negative (N)**: Observation is not positive (eg. **is not** a dog).
- **True Positive (TP)**: Outcome where the model correctly predicts the positive class.
- **True Negative (TN)**: Outcome where the model correctly predicts the negative class.
- **False Positive (FP)**: Also called a **type 1 error**, an outcome where the model incorrectly predicts the positive class when it is actually negative.
- **False Negative (FN)**: Also called a **type 2 error**, an outcome where the model incorrectly predicts the negative class when it is actually positive.

2.2 Review of Existing Literature

There are different algorithms and methodologies for identifying the disease on plant leaves. There are many different organizations and many researchers who have studied and have done work on this topic using different algorithms. Some of them are summarized below: -

2.2.1 Yashpal Sen, Chandra Shekar Mithlesh, Dr. Vivek Baghel [1] describes an approach for disease detection of crop for economic growth of rural area. This paper discussed about an automated system for identifying and classifying different diseases of the contaminated plants is an emerging research area in precision agriculture. This paper describes the approach to prevent the crop from heavy loss by careful detection of diseases. The region of internet is leaf because most of the diseases occur in leaf only. Histogram equalization is used to pre-process the input image to increase the contrast in low contrast image, K-mean clustering algorithm which classifies objects. Disease in crop leaf is detected accurately using image processing technique it is used to analyse the disease which will be useful to farmers.

2.2.2 K. Elangoran, S. Nalini [2] presented a concept of plant disease classification using image segmentation and SVM techniques. This paper describes an image processing technique that identifies the visual symptoms of plant diseases using an analysis of coloured images, work of software program that recognizes the colour and shape of the leaf image. LABVIEW software was used to capture the image of plant RGB colour model and MATLAB software is used to enable a recognition process to determine the plant disease through the leaf images. The colour model respectively was used to reduce effect of illumination and distinguish between leaf colours efficiently and the resulting colour pixels are clustered to obtain groups of colours in the images.

2.2.3 Sandesh Raut, Karthik Ingale [3] proposed fast and accurate method for detection and classification of plant diseases. The proposed algorithm is tested on main five diseases on the plant they are Early Scorch, Cottony Mold, Ashen Mold, Late scorch, Tiny Whiteness. Initially the RGB image is acquired then a colour transformation structure for the acquired RGB leaf image is created. After that colour value in RGB converted to the space specified in the colour transformation structure. In the next step, the segmentation is done by using K-means clustering

technique after that mostly green pixels are masked. Finally, the feature extracted was recognized through a pre-trained neural network. The result show that the proposed system can successfully detect and classify the diseases with a precision between 83% and 94%.

2.2.4 Sagar Patil, Anjali chandavale [4] this survey mainly concentrates on disease detection of dicot plants, here the image acquisition is done by taking RGB image pattern as input and transform it into HSI form, after that for texture analysis CCM and SGDM is used. In agricultural field, rice cultivation plays a vital role. But their growths are affected by various diseases. There will be decrease in the production if the diseases are not identified at an early stage. The main goal of this work is to develop an image processing system that can identify and classify the various rice plant diseases affecting the cultivation of rice namely brown spot disease, leaf blast diseases and bacterial blight disease. This work can be divided into 2 parts namely-rice plant disease detection and recognition of rice plant diseases. In disease detection, the disease affected portion of the rice plant is first identified using KNN and clustering classifier. After that, in disease recognition the rice plant disease type is recognized using classifiers namely KNN and SVM.

2.2.5 T. Rumpf, A-K , Mahlein, U.Steiner , E-C Oerke[5] Work expertise in plant diseases requires successive processing time hence image processing is used for the detection of plant diseases, this paper discuss the method used for the detection of plant diseases using the leaves images, and also various techniques to segment the disease part of the plant this paper also discussed some feature extraction and classification techniques to extract the features of infected leaf and the classification of plant diseases the accurately detection and classification of the plant diseases is very important for the successful cultivation of Crop and this can be done using image processing the use of an n methods for classification of disease in plant such as self-organizing feature map block back propagation algorithm SVM excreta can be efficiently used from these methods we can accurately identify and classify various plant diseases using image processing techniques machine learning methods such as artificial neural network decision tree Cayman neighbour's and support vector machine have been applied in Agricultural Research this paper also discussed dichotomous classification between healthy leaves and leaves with disease symptoms the results showed that the specificity of the classification was always lower than the

sensitivity the classification error range was 7% to almost 3% the classification accuracy increased with increasing disease severity the result showed that the classification accuracy was about to 65% for all diseases.

2.2.6 Mr. Sanjay Mirchandani, Mihir pendse, Prathamesh Rane, Ashwini Vedula[6] researchers proposed detection and classification of plant disease using image processing and artificial neural networks in this paper a software solution for fast accurate and automatic detection and classification of plant disease through image processing identification of disease is key to preventing losses in the quality and quantity of the agricultural product this paper discussed the detection and classification of plant diseases is divided into three steps such as identification of infected object extraction of feature set of the infected leaf images and detection and classification the type of disease using ANN. Different techniques are adopted for detection and diagnosis the disease but the better way is by image processing the author suggested a method in which initially the infected region is found then different features are extracted such as colour texture and shape finally parameter classification technique is used for detecting the diseases.

2.2.7 Savita N Ghaiwat, Parul Arora[7] presented an image processing technique for detection and classification of plant disease classification technique deals with classifying each pattern in one of the distinct classes the author suggested so many techniques for classification such as K nearest neighbour classifier probabilistic neural network genetic algorithm support vector machine and principal component analysis artificial neural network Fuzzy Logic the technique is presented using image processing as a tool to enhance the feature extraction of an image by using of local binary pattern as a parameter the local binary pattern approach has evolved to represent a significant breakthrough in texture analysis outperforming earlier method in many application study of image analysis takes which have not been generally considered texture analysis problems is done results suggest that texture analysis and the ideas behind the lbp method could have a much wider role in image analysis and computer vision then was thought before.

2.2.8 Amar Kumar dey [8] This paper deals with leaf rot disease detection for betel vine based on image processing. The proposed methodology has three vital stages the initial stage was the

image acquisition stage through which the real-world sample is recorded in the digital form using flatbed digital scanner next stage is image processing segmentation classification and leaf area calculation. Here 21*30 sq. cm image is Canon under flatbed scanner during test phase it acquired a series of 12 colour images using scanner and then the colour images were digitalized at a resolution of 300 DPI to produce RGB digital colour images. The digital version of the leaf sample consists of a about 30% of leaf area and rest 70% is background in order to achieve fast processing digital image of leaf cropped into a smaller dimension of size after this is a state the digital version of the sample leaf image consist about 70% of leaf area part and rest 30% as background colour feature are used to identify the affected area. RGB, hsv, ycbcr colour models are used to colour identification these three channels result in 12 individual images then the queue is responsible for masking and approximated threshold value is applied to affected leaf was calculated.

2.2.9 Jayamala k Patil, Rajkumar [9] To retrieve the related images the search is done in Two Steps, first step is matching the images by comparing the standard deviations for three colour components the second step is weighted version of the Euclidean distance between the feature coefficients of an image selected in the first step, this reported following important image processing method first one image clipping separating the leaf with spots from the complex background. Noise resolution to filters simple filter and median filter work compared and at last median filter was chosen. Thresholding to segment or partition image into the spot background fourth one segmentation they used OSTU's method. K-means clustering and back propagation feed-forward neural network for clustering and classification. There are two main characteristics of plant disease detection using machine learning method that must be achieved their speed and accuracy hence innovative efficient and fast interpreting algorithms which will help plants scientists in detecting disease work proposed by the researcher can be extended for development of hybrid algorithms such as genetic algorithms and neural networks in order to increase the recognition rate of the final classification process.

2.2.10 S Arivazhagan, R Newlin shebiah, S Ananthi, S Vishnu varthini [10] the approaches and methodologies which are used in this survey includes RGB acquisition: input image and consider the image colour according to the RGB model, Colour transformation structure:

includes the transformation of colours from RGB to hsv (Hue saturation intensity) h component taken into analysis. Masking green pixels: It identifies the mostly green coloured pixels based on threshold value. Segmentation: Infected portion of the leaf is extracted and divide them into patches, and extract the useful segments. CCM texture analysis is developed through sgdm. Classifier: as a classifier minimum distance criterion is used and SVM are used for better classification and regression. Hear the application of texture analysis is highlighted by this technique only some group of plant disease can be detected and SVM has introduced in less computation method.

2.2.11 A.Singh, B.Ganapathysubramanian, A.K.Singh, S.Sarkar[11] The aim of this paper is to give us an overview regarding the work done in the field of plant stress phenotyping using Machine Learning, classification, quantification and prediction. It will also tell about the general issues in Machine Learning strategy.

Techniques/Methodology adopted in paper:

• High-throughput phenotyping (HTP), High-throughput stress phenotyping (HTSP), ML algorithms, Support vector machines (SVM), Artificial Neural Networks (ANN)

Findings/Results:

This review gave us an overview of Machine Learning and with the various advantages of machine learning in the future. The concepts discussed here can be applied to data collected across the spectrum of complexity and sophistication. We have identified several futures avenues for using ML techniques that show tremendous promise but remain currently unutilized by the phenotyping community.

2.2.12 A.Camargo, J.S.Smith[12] processing from the visual symptoms by analysing the coloured images.

Techniques/Methodology adopted in paper:

• Image pre-processing >Image enhancement > Image segmentation > Image post-processing

Findings/Results:

• The test set consisted of 20 images which were showing symptoms of plant disease in different crops used in the study. To create the manually segmented set of images, a grid was overlaid on

the image and each position was then evaluated the white colour and black colour. White colour (1) depicted the pixel having diseased symptoms whereas the black (0) for non-diseased region.

• To evaluate the algorithm, original images were automatically segmented. The output which was produced was a binary image where 1 represented a pixel classified as diseased and 0 as non-diseased.

2.2.13 VijaiSingh, A.K.Misra[13] The aim of this paper we do the automatic detection and classification of plant leaf diseases using an algorithm for image segmentation technique. It also covers survey on different diseases classification techniques that can be used for plant leaf disease detection.

Techniques adopted in paper:

Genetic algorithm, k-nearest-neighbor method, Machine learning based recognition, Colour Co-occurrence Method, Artificial neural network (ANN)

Findings/Results:

• By using Minimum Distance Criterion with K-Mean Clustering we did the first classification which showed its efficiency and accuracy of 86.54%. The detection accuracy was later improved to 93.63% by the proposed algorithm.

• The second phase of classification was done using SVM classifier and shows efficiency and accuracy of 95.71%. But with the help of proposed algorithm, we were able to improve the detection accuracy to 95.71%.

• From the results it was clearly visible that only few samples from the Frog eye leaf spot and bacterial leaf spot leaves were misclassified. Only two leafs with bacterial leaf spot disease was classified as frog eye leaf spot and one frog eye leaf spot was classified as the bacterial leaf spot. The average accuracy of proposed algorithm is 97.6%.

2.2.14 S.Phadiar ,J.Sil[14] The aim of this paper is to describe a software prototype system for the detection of disease in rice plant on the basis of various images of the rice plants. Images of the infected part of the rice plant is taken using digital camera. In order to detect the defected part of the plant various techniques like image segmentation, image growing etc. have been used. By using neural network the infected part of the leaf is classified. Image processing and soft computing techniques have been applied on infected rice plant.

Techniques/Methodology adopted in paper:

• Image processing and pattern analysis methods, Hue Intensity Saturation (HIS) model, Bi-level thresholding method, Boundary detection algorithm using 8- connectivity method, Self organizing map (SOM)

Findings/Results:

• In this research paper, the infected part of the rice plant is being classified using SOM (Self Organizing Map) neural network where the images are being obtained by doing the extraction of the infected part while four different types of images are being used for the testing purposes.

• Usage of zooming algorithm is also there for feature extraction of the image. Zooming algorithm by the usage of computationally efficient technique extracts the features of the image.

2.2.15 A. Meunkaewjinda, P. Kumsawat and K. Attakitmongcol [15] The aim of this paper is to do automatic plant disease diagnosis with the usage of multiple artificial intelligent techniques. In this paper the main focus is on the grape leaf disease. Once the system is trained, it can diagnose the plant leaf disease without doing its maintenance again, and again from the beginning.

Techniques/Methodology adopted in paper: Genetic algorithm for optimization, Support vector machines for classification, Artificial neural network, Back-propagation neural network (BPNN, Anisotropic diffusion technique, Modified self-organizing feature map (MSOFM)

Findings/Results:

• Grape leaf disease extraction and classification system using colour imagery, the system gives the desirable results. Back-propagation neural network provides efficient grape leaf extraction with complex background where as Modified self-organizing feature map and Genetic algorithm provides automatic adjustment for the colour extraction of the diseased grape leaf.

• This system has tested images of 426x568 pixels. There were 497 scab disease samples, 489 rust disease sample and 492 non disease samples used to train the SVMs. For testing stage there were 39 scab disease images,41 rust disease images and 35 non-disease images. The results show that the system provides desirable performance.

2.2.16 E.Omrani, B.Khoshnevisan, S.Shamshirband, H.Saboohi, N.B.Anuar, M.H.N.M.Nasir [16] The aim of this paper is to classify disease using soft computing

approaches, Artificial Neural Networks (ANNs), Support Vector Machines (SVM) in apple. Techniques/Methodology adopted in paper:

• K-means clustering, Artificial neural networks (ANNs), Support vector machines (SVMs), Gray level co-occurrence matrix (GLCM), Polynomial based (SVR_Poly), RBF-based SVR (SVR_rbf), Wavelet transforms, Principal component analysis (PCA), Back-propagation neural network.

Findings/Results:

• The usage of K-means and ANNs for clustering and classifying diseases affecting the leaves of plant. The outcome is 94% correct and swifter by 20%.

• K-means cluster divided the images into two groups: infected and healthy leaf areas. The diseases were classified based on the extracted features.

• The SVR_rbf model had a very small RMSE (0.13) during training and the value was 0.2 in testing.

• The SVR_poly had RMSE of 0.39 in training and RMSE of 0.42 during testing. It was seen that SVR_rbf model showed consistently good correlation throughout training and testing.

• A comparison of SVR_rbf results with SVR_poly and ANN reveals that SVR_rbf outperforms the POLY model in terms of prediction accuracy.

2.2.17 S.Bashir, N.Sharma [17] Disease can be recognized by using colour and texture features. Disease detection in Malus Domestica is using K mean clustering, colour and texture analysis.

Techniques/Methodology adopted in paper:

SVM Segmentation, K-mean clustering, Back propagation, Neural Network, Co-occurrence matrix method

Findings/Results:

• Appropriate enhancement of images uses histograms. Image segmentation is used for presence of adequate symptoms for detection of disease.

• Spot on the image can be detected by texture segmentation. Rough, silky, bumpy texture of image can be identified by texture analysis. Texture analysis uses co-occurrence matrix method, which uses Hue Saturation Intensity colour space representation.

• Colourfulness in HIS space is given by the saturation component and transformation of colour space can be done easily.

2.2.18 J.G.A.Barbedo [18] Detecting, quantifying and classifying plant diseases using digital image processing techniques.

Techniques/Methodology adopted in paper:

Neural Networks, Thresholding, Dual-segmented regression analysis, Quantification, Colour analysis, Fuzzy Logic, Knowledge-based system, Sobel operator, Chlorosis algorithm

Findings/ Results:

• Background is discriminated from the leaf and then damaged regions is separated from healthy surface and the ratio between the number of pixels in damage by the total number of pixels of the leaf gives the final estimate.

• Subsequent steps use 2 modification versions of I3 and only one modification of H. Separation of diseased and healthy region is done using thresholding.

• Red and green component of image are combined using chlorosis algorithm for determining the yellowness of leaf. To discriminate leaves from background blue component is used. To identify and quantify the necrotic region is done Necrosis algorithm.

• Area occupied by the spots is estimated using thresholding the blue component of the image and algorithm to implement using white spots algorithm.

2.2.19 F.Qin, D.Liu, B.Sun, L.Ruan, Z.Ma, H.Wang [19] Using Image recognition technology identification of Alfalfa leaf diseases.

Techniques/Methodology adopted in paper:

Fuzzy *C*-means clustering, *k*-median clustering, Euclidean distance, Logistic regression analysis, Naive bayes algorithm, linear discriminant analysis

Findings/ Results:

• Arithmetic square root of total number of features was randomly selected by each decision tree. For e.g. If arithmetic square root is decimal, then rounding up the decimal gives the number of features randomly selected by each decision tree.

• Disease recognition models built after feature selection gives the satisfactory recognition results. This indicates that features extracted from lesion images were efficient.

• For implementing K-median clustering algorithm linear discriminant was used and the highest score of median and mean are used for implementation.

2.2 SUMMARY					
S.NO	PAPER NAME	AUTHOR	OBJECTIVE	METHODOLOGY	FINDINGS
1.	Detection of plant leaf diseases using image segmentation and soft computing techniques	Vijai Singh et.al	To monitor the crop growth using the image segmentation techniques.	· Support Vector Machine · Artificial Neural Network · Dispersion method · Self-sorting out element	The information for ripening stages of crop and infected part recognition is made.
2.	Remote Area Plant Disease Detection Using Image Processing	Sabah Bashir et.al	Infected part in the plant can be detected with help of color, and other changing properties by using classification algorithm.	· Segmentation · RGB · Colour transformation · Image acquisition · classification	Different pixel information is extracted and Green leaves pixel and diseased leaf pixel are compared by finding the ratio of pixel corresponding to the healthy leaf to the pixel corresponding to the infected leaf
	Smart Farming: Pomegranate Disease Detection Using Image Processing	Manisha Bhangea et.al	The paper provides the image processing techniques and the algorithm which help the farmers tp identify the disease in pomegranate.	· Image pre-processing · Feature extraction · Morphology · Colour Coherence Vector (CCV) · Clustering Training and classification	Remote area farmer can identify the disease in the pomegranate crop.
4.	Machine Learning for	Arti Singh et.al	To give us an overview	• High throughput phenotyping (HTP)	The concepts discussed here

	High-Throughput Stress Phenotyping in Plants		regarding the work done in the field of plant stress phenotyping using Machine Learning, classification, quantification and prediction.	• High-throughput stress phenotyping (HTSP). • ML algorithms • Support vector machines (SVM) • Artificial Neural Networks(ANN)	can be applied to data collected across the spectrum of complexity and sophistication.
5.	Image pattern classification for the identification of disease causing agents in plants	A.Camargo et.al	To do the automatic identification of the plant disease by image processing from the visual symptoms by analyzing the colored images.	• Image pre-processing •Image enhancement •Image segmentation •Image post-processing	The test set consisted of 20 images which were showing symptoms of plant disease in different crops used in the study.
6.	Leaf Disease Detection using Image Processing.	Sujatha R et.al	This study summarizes major image processing used for identification of leaf diseases are k-means clustering, SVM.	•Image Acquisition •Image pre-processing •Image segmentation •Feature Extraction in Image •Detection And Classification of the diseases	By using this concept the disease identification is done for all kinds of leafs and also the user can know the affected area of leaf in percentage by identifying the disease properly the user can rectify the problem very easy and with less cost.

7.	Leaf Disease Detection using Image Processing.	Ashwini C et.al	This project proposed to point out disease in the leaf with a union of shape, texture and color feature withdrawal.	•Image Acquisition •Image Sharpening •Edge Detection •Greyscale Image •Adaptive Histogram Equalization •Shape, Color and Texture Features •Support Vector Machine(SVM)	Initially the farmers sends a digital image of the diseased leaf of a plant and these images are read in MATLAB and processed automatically based on SVM.
8.	Plant diseases and pests detection based on deep learning.	Jun Liu et.al	Plant diseases and pests identification can be carried out by means of digital image processing.	•Using network as feature extractor •Using network for classification directly •Using network for lesions location •Detection network	Deep learning technology has made some achievements in the identification of plant diseases and pests. Various image recognition algorithms have also been further developed and extended, which provides a theoretical basis for the identification of specific diseases and pests.
9.	Using Deep Learning for Image Based Plant Disease Detection	Sharada P.Mohanty et.al	This project proposed to point out disease in the leaf with a union of shape, texture and color feature withdrawal.	•Leaf disease detection by well-known deep learning Architecture. •New/modified dl architectures for leaf-disease detection.	The approach of training deep learning models on increasingly large and publicly available image datasets presents a clear path toward

				•The system of leaf-disease detection	smartphone-assisted crop disease diagnosis on a massive global scale.
10.	Rice disease detection Pattern Recognition techniques	Minu Eliz Pothen et.al	To depict a product model framework for the discovery of malady in rice plant based on different pictures of the rice plants.	•Preparing & design examination strategies of images •Binary cut off methods •Border layout calculation using eight-availability strategy •Self-organizing map(SOM)	The diseased part of the rice plant leaf is identified with the help of the self-organizing map. Testing is done using four different images of the crop. Infected region is extracted using neural networks pattern recognition techniques.
11.	Smart Plant Disease Detection System	Bhoopendra Joshi et.al	The paper deals with association between disease symptoms and impact on product yield.	•Data Preprocessing •CNN Algorithm	This paper gives the survey on different diseases classification techniques. Tomato, Potato, Pepper-bell are some of those species on which the algorithms and methods were tested.
12.	Plant disease detection using image	Sachin D. Khirade et.al	To provide different types image processing	•Image pre processing •Image enhancement	The feature extraction and the image segmentation

	processing.		techniques which can directly be implemented in MATLAB for preprocessing of the image and also brief about the image segmentation and image classification.	•SVM classification •Semantic networks •K means clustering •Neural network based classifier	algorithm used in this paper are efficient with very high accuracy. Different disease are identified with very high precision rate and accuracy. Clustering algorithm approach is also very much efficient fast, the clustering algorithm segment the image in the different clusters in very short span of time .
13.	Comprehensive Survey on various Plant Disease Detection Techniques based on Machine Learning	Anchal Chaudhary et.al	To calculate and evaluate the amount of infectious part and determines the change in shade of that affected part.	• Plant diseases recognition is the mechanism having three stages: • The images are fed into the terminal for pre-processing. • The images are analyzed according to the segmentation features. • Classification of the picture is performed by employing some desired	In this paper they calculate the digital image processing for the detection, diagnosis, reorganization of the crop leaf infections based on the usage of the k-means clustering technique and it also increases the accuracy.

				classification model.	
14.	An Overview of the Research on Plant Leaves Disease detection using Image Processing Techniques.	Ms. Kiran R. Gavhale et.al	The objective of this paper is to concentrate on the plant leaf disease detection based on the texture of the leaf. Leaf presents several advantages over flowers and fruits at all seasons worldwide.	• RGB image acquisition • Convert the input image into color space • Segment the components • Obtain the useful segments • Computing the texture features • Configuring the neural networks for recognition	These techniques are used to analyses the healthy and diseased plants leaves.
15.	Detection of unhealthy region of plant leaves and classification of plant leaf diseases using texture features	Arivazhagan Selvaraj et.al	Detection and classification of leaf diseases has been proposed.	• HSI • Colour co-occurrence matrix • texture features • SVM	An application of texture analysis in detecting and classifying the plant leaf diseases has been explained in this paper. The experimental results indicate the proposed approach can recognize and classify the leaf diseases with a little computational effort.

Chapter No. 3

RESEARCH DESIGN & METHODOLOGY

3.1 System Design

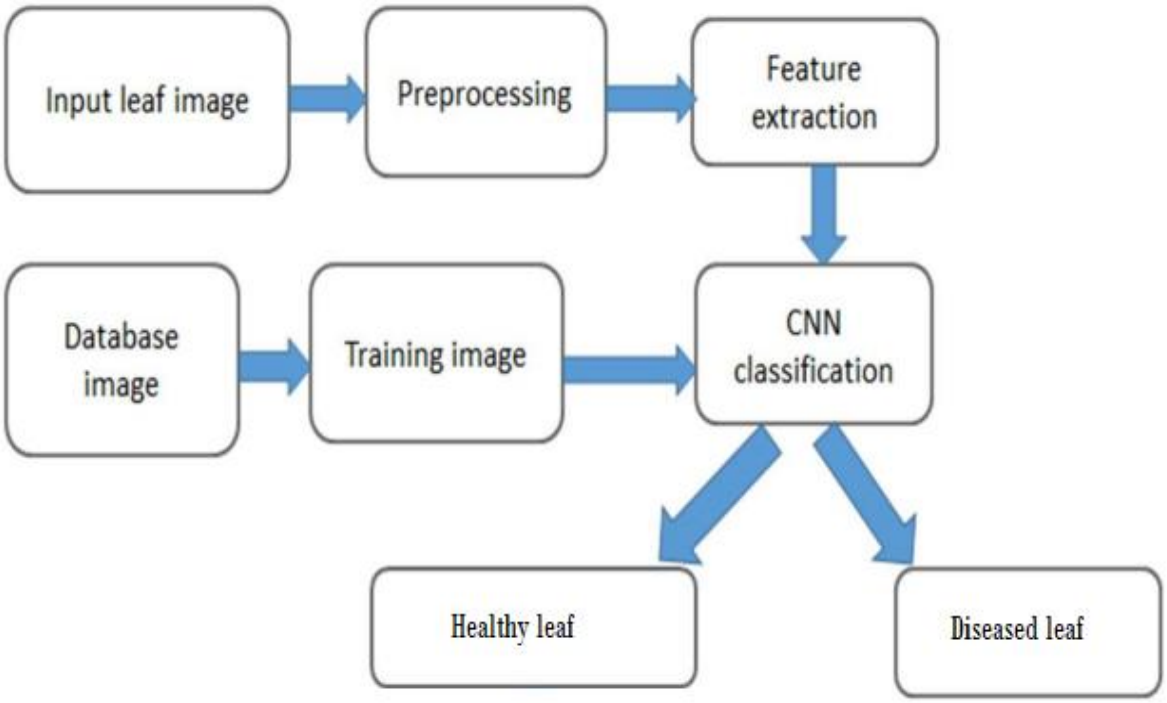

Fig 3.1 – *System Block Diagram*

This system block diagram illustrates the main modules involved in our project.

For the purpose of image-based identification which includes, training phase to evaluation phase where the performance of classification algorithms are evaluated, it is necessary to have huge datasets.

Pre-processing images involves outsourcing background noise, intensity normalization of individual image particles, removing reflections and masking portions of images.

The dataset of the plant leaf images are trained in python tool and classified into various clusters which classifies various labels. Convolution Neural Networks is designed for accurate analysis. Based on the feature extraction parameters, the algorithm predicts whether the leaf is healthy leaf or diseased one.

3.2 Detailed Design

3.2.1 Sequence Diagram

Sequence diagrams are sometimes called event diagrams or event scenarios. A sequence diagram shows as parallel vertical lines (lifelines), different processes or objects that live simultaneously and as horizontal arrows, the messages exchanged between them in the order in which they occur.

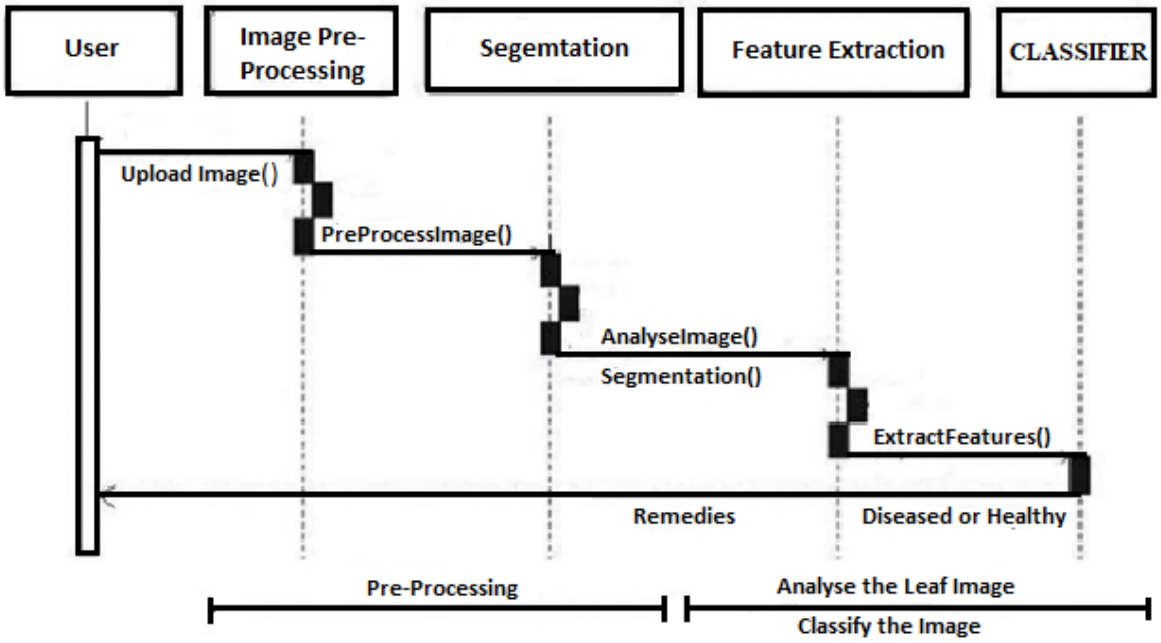

Fig 3.2.1 – *Sequence Diagram*

1. Usage scenarios: A usage scenario is a description of a potential way your system is used. The logic of a usage scenario may be part of a use case, perhaps an alternate course. It may also be one entire pass through a use case, such as the logic described by the basic course of action or a portion of the basic course of action, plus one or more alternate scenarios. The logic of a usage scenario may also be a pass through the logic contained in several use cases.

2. The logic of methods: Sequence diagrams can be used to explore the logic of a complex operation, function, or procedure. One way to think of sequence diagrams, particularly highly detailed diagrams, is as visual object code.

3. The logic of services: A service is effectively a high-level method, often one that can be invoked by a wide variety of clients. This includes web-services as well as business transactions implemented by a variety of technologies.

3.1.2 Activity Diagram

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as in operation of the system. The control flow is drawn from one operation to another.

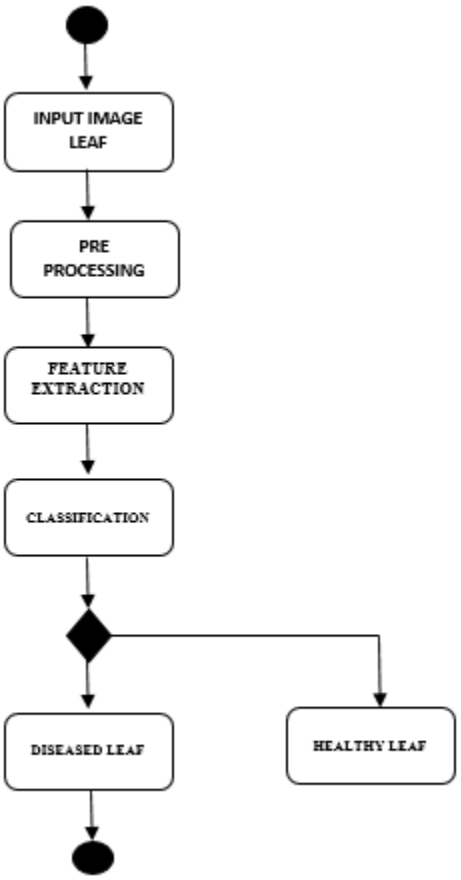

Fig 3.2.2 – *Activity Diagram*

The above flow represents the flow from one activity to another activity. The activity starts from input leaf image through digital camera, and then input leaf is pre-processed. The features like colour, shape, texture, etc. are extracted. Then, the processed image is classified as Normal or Abnormal, if Abnormal is found in the leaf, then remedies will be suggested.

Chapter No. 4

IMPLEMENTATION & RESULT ANALYSIS

4.1 Dataset

This is a set of images for specified purposes. We use a plant leaf dataset and each of them is divided for preprocessing and classification. The Leaf dataset consists of 4300 leaf images with 3600 images for training dataset and 700 images for validation dataset. This contains both healthy and diseased leaves.

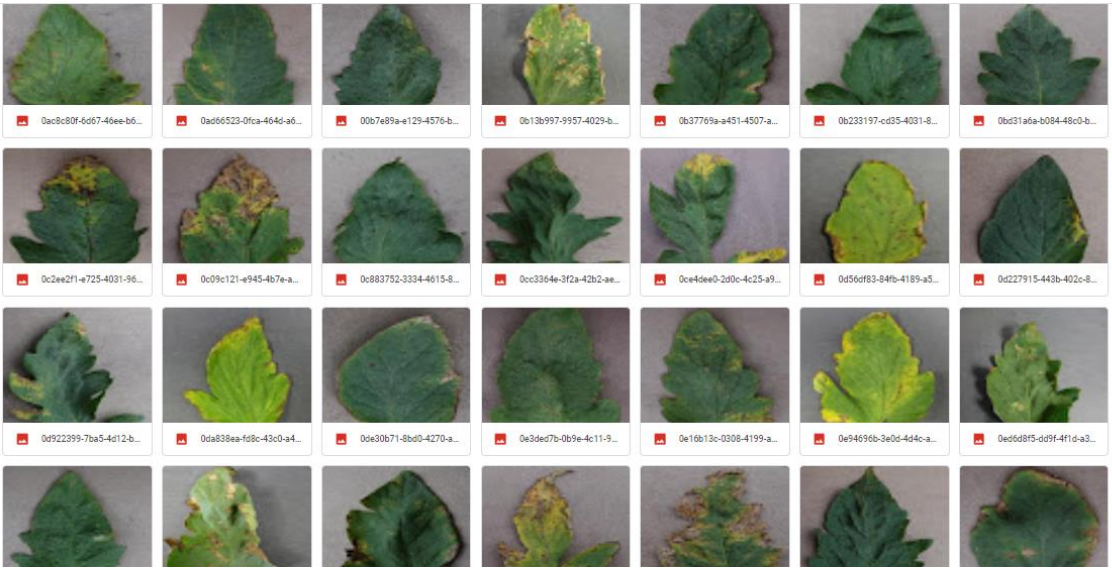

Fig 4.1.1 Training Dataset (Disease Leaf)

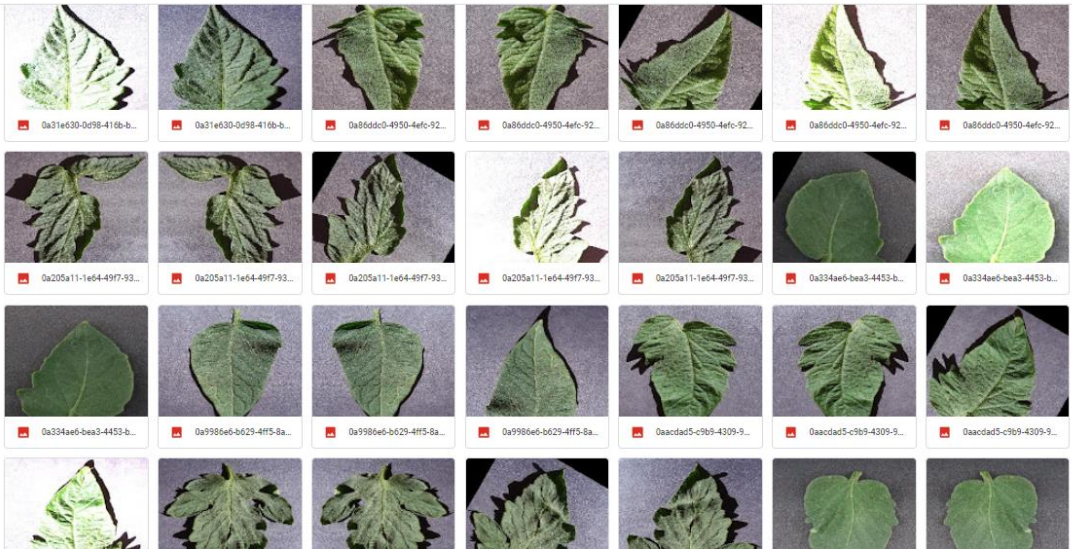

Fig 4.1.2 *Training Dataset (Healthy Leaf)*

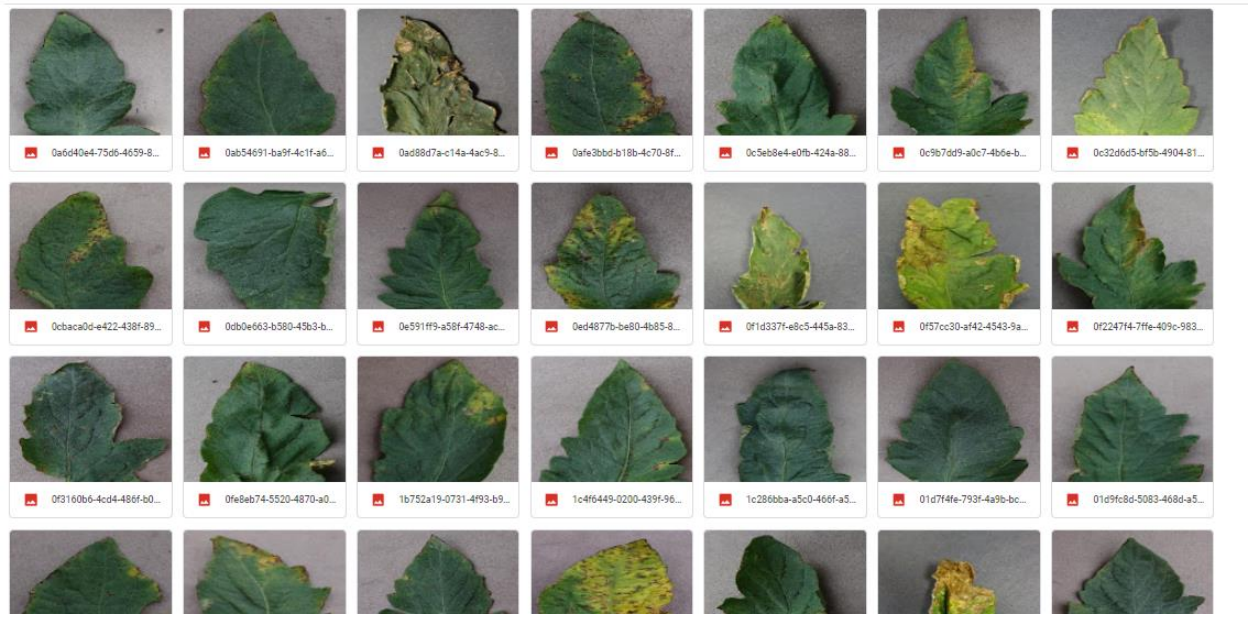

Fig 4.1.3 Validation Dataset (Diseased Leaf)

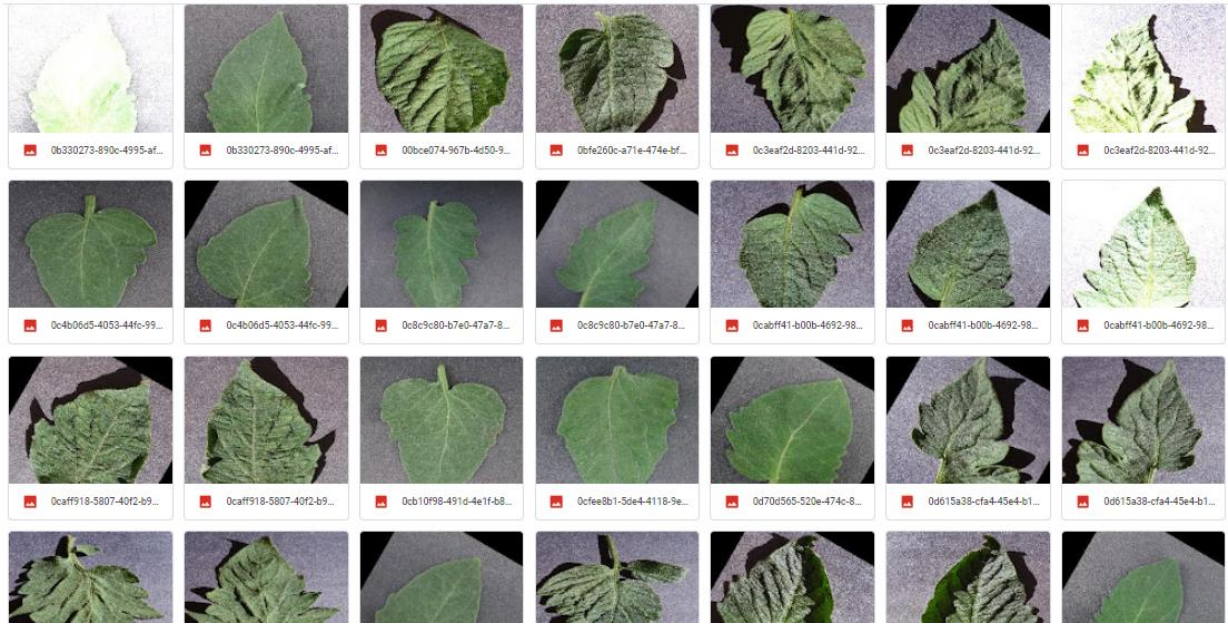

Fig 4.1.4 Validation Dataset (Healthy Leaf)

4.2 Preprocessing

The database is preprocessed such as Image reshaping, resizing and conversion to an array form. Similar processing is also done on the test image.

4.3 Model training and Testing

The training dataset is used to train the model (CNN) so that it can identify the test image and the disease it has. CNN has different layers that are Dense, Dropout, Activation, Flatten, Convolution2D, and MaxPooling2D. After successful training and preprocessing, comparison of the test image and trained model takes place to predict the disease.

Importing required Libraries and Mounting to drive

```
from google.colab import drive
drive.mount("/content/drive/")
import tensorflow as tf
import os
import glob
import skimage
import random
import pandas as pd
import numpy as np
import seaborn as sns
import keras
from keras.layers import Dense, Conv2D , MaxPooling2D, Activation , Flatten , Dropout , Batch Normalization
from sklearn.metrics import classification_report,confusion_matrix
import math
import cv2
from PIL import Image
import numpy as np
from keras import layers
from keras.callbacks import Callback, ModelCheckpoint, ReduceLROnPlateau, TensorBoard
from keras.preprocessing.image import ImageDataGenerator
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from keras.models import Model
from sklearn.model_selection import train_test_split
from sklearn.metrics import cohen_kappa_score, accuracy_score
import scipy
from tqdm import tqdm
from keras import backend as K
import cv2 as cv
import gc
from functools import partial
from sklearn import metrics
from collections import Counter
import json
from tensorflow.keras.applications import EfficientNetB7
import itertools
```

```
import matplotlib.pyplot as plt
%matplotlib inline
from skimage import io as io
```

Loading the dataset

```
def Dataset_loader(DIR, RESIZE, sigmaX=10):
    IMG = []
    read = lambda imname: np.asarray(Image.open(imname).convert("RGB"))
    for IMAGE_NAME in tqdm(os.listdir(DIR)):
        PATH = os.path.join(DIR,IMAGE_NAME)
        _, ftype = os.path.splitext(PATH)
        img = read(PATH)
        IMG.append(np.array(img))
    return IMG

disease_train = np.array(Dataset_loader('/content/drive/MyDrive/data/train/disease',512))
healthy_train =np.array(Dataset_loader('/content/drive/MyDrive/data/train/healthy',512))
disease_test =np.array(Dataset_loader('/content/drive/MyDrive/data/valid/disease',512))
healthy_test = np.array(Dataset_loader('/content/drive/MyDrive/data/valid/healthy',512))
```

```
100%|██████████| 1800/1800 [00:20<00:00, 86.48it/s]
100%|██████████| 1800/1800 [00:20<00:00, 87.69it/s]
100%|██████████| 350/350 [00:02<00:00, 143.47it/s]
100%|██████████| 350/350 [00:02<00:00, 132.46it/s]
```

Creating numpy array of zeros for labeling healthy images and numpy array of ones for labeling diseased images. In the latter part the dataset is shuffled, labels are coverted into categorical format.

```
disease_train_label = np.zeros(len(disease_train))
healthy_train_label = np.ones(len(healthy_train))
disease_test_label = np.zeros(len(disease_test))
healthy_test_label = np.ones(len(healthy_test))

# Merge data
X_train = np.concatenate((disease_train, healthy_train), axis = 0)
Y_train = np.concatenate((disease_train_label, healthy_train_label), axis = 0)
X_test = np.concatenate((disease_test, healthy_test), axis = 0)
Y_test = np.concatenate((disease_test_label, healthy_test_label), axis = 0)

# Shuffle train data
s = np.arange(X_train.shape[0])
np.random.shuffle(s)
X_train = X_train[s]
Y_train = Y_train[s]
```

```
# Shuffle test data
s = np.arange(X_test.shape[0])
np.random.shuffle(s)
X_test = X_test[s]
Y_test = Y_test[s]

# To categorical
Y_train = to_categorical(Y_train, num_classes= 2)
Y_test = to_categorical(Y_test, num_classes= 2)
print(X_train.shape)
print(Y_train.shape)
print(X_test.shape)
print(Y_test.shape)
```

```
(3600, 256, 256, 3)
(3600, 2)
(700, 256, 256, 3)
(700, 2)
```

```
x_train = X_train
y_train =Y_train
x_test = X_test
y_test =Y_test
x_train, x_test, y_train, y_test = train_test_split(
   X_train, Y_train,
  test_size=0.2,
   random_state=12,
)
print(x_train.shape)
print(y_train.shape
print(x_test.shape)
print(y_test.shape)
```

```
(2880, 256, 256, 3)
(2880, 2)
(720, 256, 256, 3)
(720, 2)
```

Create the model

```
model = Sequential()
model.add(Conv2D(16,(5,5),padding='valid',input_shape=(256, 256, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding = 'valid'))
model.add(Dropout(0.4))
model.add(Conv2D(64,(5,5),padding='valid'))
```

```
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding = 'valid'))
model.add(Dropout(0.6))
model.add(Conv2D(64,(5,5),padding='valid'))
model.add(Activation('relu'))
model.add(Dropout(0.8))
model.add(Flatten())
model.add(Dense(256, name="dense_1"))
model.add(Activation('softmax'))
model_feat = Model(inputs=model.input,outputs=model.get_layer('dense_1').output)
feat_train = model_feat.predict(x_train)#Extracted Features from CNN
feat_test = model_feat.predict(x_test)
# print(feat_test)

print(feat_train)
```

```
[[ -0.3978734   22.809122    -8.117481   ...  18.022316     3.4125786
   -0.39403152]
 [ -3.1041985   17.11131      1.7258587  ...  11.737001     6.3713045
  -18.214457  ]
 [  1.9909501   13.334702    -3.7272692  ...  10.676624    11.523933
  -12.691799  ]
 ...
 [ -5.3437314   16.679964    -5.6085577  ...   5.3486543    4.211545
  -16.119999  ]
 [ -8.690926    30.297922    -7.468353   ...   9.9811      10.948505
   -4.5574045 ]
 [-16.574469    32.233498    -9.869247   ...   9.052207    15.516136
  -20.722013  ]]
```

Gaussian Radial Basis Function (RBF) kernel for SVM

```
from sklearn.svm import SVC
svclassifier = SVC()
svclassifier = SVC(kernel='rbf', random_state=42,C=1.0, degree=3,
        gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200,
class_weight=None, verbose=False,
        max_iter=-1, decision_function_shape='ovr', break_ties=False)
svclassifier.fit(feat_train,np.argmax(y_train,axis=1))
y_pred = svclassifier.predict(feat_test)

from sklearn.metrics import accuracy_score
print('Accuracy: %.3f' % accuracy_score(np.argmax(y_test,axis=1), y_pred))
from sklearn.metrics import precision_score
print('Precision: %.3f' % precision_score(np.argmax(y_test,axis=1), y_pred))
from sklearn.metrics import recall_score
print('Recall: %.3f' % recall_score(np.argmax(y_test,axis=1), y_pred))
from sklearn.metrics import f1_score
print('F1 Score: %.3f' % f1_score(np.argmax(y_test,axis=1), y_pred))
```

```
Accuracy: 0.958
Precision: 0.976
Recall: 0.938
F1 Score: 0.957
```

Naive Bayes Classifier

```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB(priors=None, var_smoothing=1e-09)
model.fit(feat_train,np.argmax(y_train,axis=1))
y_pred=model.predict(feat_test)
from sklearn.metrics import accuracy_score
print('Accuracy: %.3f' % accuracy_score(np.argmax(y_test,axis=1), y_pred))
from sklearn.metrics import precision_score
print('Precision: %.3f' % precision_score(np.argmax(y_test,axis=1), y_pred,))
from sklearn.metrics import recall_score
print('Recall: %.3f' % recall_score(np.argmax(y_test,axis=1), y_pred))
from sklearn.metrics import f1_score
print('F1 Score: %.3f' % f1_score(np.argmax(y_test,axis=1), y_pred))
```

```
Accuracy: 0.750
Precision: 0.787
Recall: 0.671
F1 Score: 0.725
```

CNN model - Efficientnetb7 model

```
def build_model(num_classes):
    base_model = EfficientNetB7(input_shape = (256, 256, 3),  weights='imagenet',include_top =
False,)
    for layer in base_model.layers:
        layer.trainable = False
    x = base_model.output
    x = Flatten()(x)
    x = Dense(256, activation="relu")(x)
    x = Dropout(0.5)(x)
    predictions = Dense(2, activation="softmax")(x)
    model = tf.keras.Model( base_model.input,predictions)
    model.compile(
        optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"]
    )
    return model
K.clear_session()
gc.collect()
model = build_model(num_classes=2)
model.summary()
```

Data Augumentation

```
BATCH_SIZE = 20
train_generator = ImageDataGenerator(
        zoom_range=2,  # set range for random zoom
        rotation_range = 90,
        horizontal_flip=True,  # randomly flip images
        vertical_flip=True,  # randomly flip images
    )
```

Training and Evaluation

```
history = model.fit(
    train_generator.flow(x_train, y_train,batch_size=BATCH_SIZE),
    steps_per_epoch=x_train.shape[0] / BATCH_SIZE,
    epochs=25,
    validation_data=train_generator.flow(x_test, y_test, batch_size=BATCH_SIZE),
    callbacks=[learn_control, checkpoint]
)
```

```
Epoch 1/25
144/144 [==============================] - ETA: 0s - loss: 5.3562 - accuracy: 0.7896
Epoch 1: val_accuracy improved from -inf to 0.88611, saving model to
/content/drive/MyDrive/efficientmodel.h5
144/144 [==============================] - 388s 3s/step - loss: 5.3562 - accuracy:
0.7896 - val_loss: 0.3050 - val_accuracy: 0.8861 - lr: 0.0010
Epoch 2/25
144/144 [==============================] - ETA: 0s - loss: 0.4842 - accuracy: 0.8580
Epoch 2: val_accuracy improved from 0.88611 to 0.94028, saving model to
/content/drive/MyDrive/efficientmodel.h5
144/144 [==============================] - 401s 3s/step - loss: 0.4842 - accuracy:
0.8580 - val_loss: 0.2783 - val_accuracy: 0.9403 - lr: 0.0010
Epoch 3/25
144/144 [==============================] - ETA: 0s - loss: 0.3359 - accuracy: 0.8806
Epoch 3: val_accuracy improved from 0.94028 to 0.95694, saving model to
/content/drive/MyDrive/efficientmodel.h5
144/144 [==============================] - 353s 2s/step - loss: 0.3359 - accuracy:
0.8806 - val_loss: 0.1737 - val_accuracy: 0.9569 - lr: 0.0010
Epoch 4/25
144/144 [==============================] - ETA: 0s - loss: 0.2815 - accuracy: 0.8913
Epoch 4: val_accuracy did not improve from 0.95694
144/144 [==============================] - 340s 2s/step - loss: 0.2815 - accuracy:
0.8913 - val_loss: 0.1469 - val_accuracy: 0.9333 - lr: 0.0010
Epoch 5/25
144/144 [==============================] - ETA: 0s - loss: 0.2880 - accuracy: 0.9031
Epoch 5: val_accuracy did not improve from 0.95694
```

```
144/144 [==============================] - 340s 2s/step - loss: 0.2880 - accuracy:
0.9031 - val_loss: 0.2176 - val_accuracy: 0.9264 - lr: 0.0010
Epoch 6/25
144/144 [==============================] - ETA: 0s - loss: 0.2857 - accuracy: 0.8934
Epoch 6: val_accuracy did not improve from 0.95694
144/144 [==============================] - 337s 2s/step - loss: 0.2857 - accuracy:
0.8934 - val_loss: 0.1409 - val_accuracy: 0.9472 - lr: 0.0010
Epoch 7/25
144/144 [==============================] - ETA: 0s - loss: 0.3840 - accuracy: 0.9010
Epoch 7: val_accuracy did not improve from 0.95694
144/144 [==============================] - 336s 2s/step - loss: 0.3840 - accuracy:
0.9010 - val_loss: 0.1784 - val_accuracy: 0.9472 - lr: 0.0010
Epoch 8/25
144/144 [==============================] - ETA: 0s - loss: 0.2559 - accuracy: 0.9118
Epoch 8: val_accuracy improved from 0.95694 to 0.96111, saving model to
/content/drive/MyDrive/efficientmodel.h5
144/144 [==============================] - 381s 3s/step - loss: 0.2559 - accuracy:
0.9118 - val_loss: 0.1116 - val_accuracy: 0.9611 - lr: 0.0010
Epoch 9/25
144/144 [==============================] - ETA: 0s - loss: 0.2361 - accuracy: 0.9139
Epoch 9: val_accuracy improved from 0.96111 to 0.96250, saving model to
/content/drive/MyDrive/efficientmodel.h5
144/144 [==============================] - 379s 3s/step - loss: 0.2361 - accuracy:
0.9139 - val_loss: 0.1841 - val_accuracy: 0.9625 - lr: 0.0010
Epoch 10/25
144/144 [==============================] - ETA: 0s - loss: 0.2340 - accuracy: 0.9128
Epoch 10: val_accuracy did not improve from 0.96250
144/144 [==============================] - 383s 3s/step - loss: 0.2340 - accuracy:
0.9128 - val_loss: 0.1396 - val_accuracy: 0.9417 - lr: 0.0010
Epoch 11/25
144/144 [==============================] - ETA: 0s - loss: 0.3044 - accuracy: 0.9205
Epoch 11: val_accuracy improved from 0.96250 to 0.97361, saving model to
/content/drive/MyDrive/efficientmodel.h5
144/144 [==============================] - 387s 3s/step - loss: 0.3044 - accuracy:
0.9205 - val_loss: 0.0909 - val_accuracy: 0.9736 - lr: 0.0010
Epoch 12/25
144/144 [==============================] - ETA: 0s - loss: 0.2698 - accuracy: 0.9014
Epoch 12: val_accuracy did not improve from 0.97361
144/144 [==============================] - 356s 2s/step - loss: 0.2698 - accuracy:
0.9014 - val_loss: 0.1153 - val_accuracy: 0.9486 - lr: 0.0010
Epoch 13/25
144/144 [==============================] - ETA: 0s - loss: 0.2273 - accuracy: 0.9108
Epoch 13: val_accuracy improved from 0.97361 to 0.97500, saving model to
/content/drive/MyDrive/efficientmodel.h5
144/144 [==============================] - 365s 3s/step - loss: 0.2273 - accuracy:
0.9108 - val_loss: 0.0935 - val_accuracy: 0.9750 - lr: 0.0010
```

```
Epoch 14/25
144/144 [==============================] - ETA: 0s - loss: 0.2229 - accuracy: 0.9094
Epoch 14: val_accuracy did not improve from 0.97500
144/144 [==============================] - 350s 2s/step - loss: 0.2229 - accuracy:
0.9094 - val_loss: 0.1160 - val_accuracy: 0.9458 - lr: 0.0010
Epoch 15/25
144/144 [==============================] - ETA: 0s - loss: 0.2169 - accuracy: 0.9149
Epoch 15: val_accuracy did not improve from 0.97500
144/144 [==============================] - 360s 2s/step - loss: 0.2169 - accuracy:
0.9149 - val_loss: 0.1016 - val_accuracy: 0.9583 - lr: 0.0010
Epoch 16/25
144/144 [==============================] - ETA: 0s - loss: 0.2153 - accuracy: 0.9205
Epoch 16: val_accuracy did not improve from 0.97500
144/144 [==============================] - 342s 2s/step - loss: 0.2153 - accuracy:
0.9205 - val_loss: 0.1246 - val_accuracy: 0.9417 - lr: 0.0010
Epoch 17/25
144/144 [==============================] - ETA: 0s - loss: 0.1816 - accuracy: 0.9264
Epoch 17: val_accuracy did not improve from 0.97500
144/144 [==============================] - 327s 2s/step - loss: 0.1816 - accuracy:
0.9264 - val_loss: 0.1380 - val_accuracy: 0.9347 - lr: 0.0010
Epoch 18/25
144/144 [==============================] - ETA: 0s - loss: 0.2018 - accuracy: 0.9306
Epoch 18: ReduceLROnPlateau reducing learning rate to 0.00020000000949949026.

Epoch 18: val_accuracy did not improve from 0.97500
144/144 [==============================] - 349s 2s/step - loss: 0.2018 - accuracy:
0.9306 - val_loss: 0.1250 - val_accuracy: 0.9583 - lr: 0.0010
Epoch 19/25
144/144 [==============================] - ETA: 0s - loss: 0.1762 - accuracy: 0.9306
Epoch 19: val_accuracy did not improve from 0.97500
144/144 [==============================] - 353s 2s/step - loss: 0.1762 - accuracy:
0.9306 - val_loss: 0.1044 - val_accuracy: 0.9667 - lr: 2.0000e-04
Epoch 20/25
144/144 [==============================] - ETA: 0s - loss: 0.1582 - accuracy: 0.9399
Epoch 20: val_accuracy did not improve from 0.97500
144/144 [==============================] - 347s 2s/step - loss: 0.1582 - accuracy:
0.9399 - val_loss: 0.0684 - val_accuracy: 0.9750 - lr: 2.0000e-04
Epoch 21/25
144/144 [==============================] - ETA: 0s - loss: 0.1271 - accuracy: 0.9538
Epoch 21: val_accuracy did not improve from 0.97500
144/144 [==============================] - 324s 2s/step - loss: 0.1271 - accuracy:
0.9538 - val_loss: 0.0823 - val_accuracy: 0.9639 - lr: 2.0000e-04
Epoch 22/25
144/144 [==============================] - ETA: 0s - loss: 0.1637 - accuracy: 0.9462
Epoch 22: val_accuracy improved from 0.97500 to 0.97917, saving model to
/content/drive/MyDrive/efficientmodel.h5
```

```
144/144 [==============================] - 345s 2s/step - loss: 0.1637 - accuracy:
0.9462 - val_loss: 0.0635 - val_accuracy: 0.9792 - lr: 2.0000e-04
Epoch 23/25
144/144 [==============================] - ETA: 0s - loss: 0.1353 - accuracy: 0.9444
Epoch 23: val_accuracy did not improve from 0.97917
144/144 [==============================] - 345s 2s/step - loss: 0.1353 - accuracy:
0.9444 - val_loss: 0.0616 - val_accuracy: 0.9750 - lr: 2.0000e-04
Epoch 24/25
144/144 [==============================] - ETA: 0s - loss: 0.1353 - accuracy: 0.9510
Epoch 24: val_accuracy did not improve from 0.97917
144/144 [==============================] - 329s 2s/step - loss: 0.1353 - accuracy:
0.9510 - val_loss: 0.0815 - val_accuracy: 0.9750 - lr: 2.0000e-04
Epoch 25/25
144/144 [==============================] - 317s 2s/step - loss: 0.1261 - accuracy:
0.9545 - val_loss: 0.0837 - val_accuracy: 0.9611 - lr: 2.0000e-04
```

Plotting the Graph

```
history_df = pd.DataFrame(history.history)
history_df[['loss', 'val_loss']].plot()
history_df = pd.DataFrame(history.history)
history_df[['accuracy', 'val_accuracy']].plot()
```

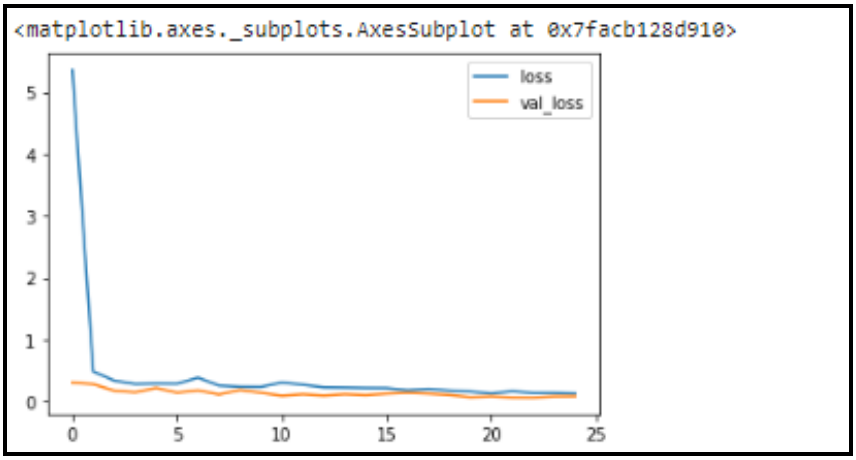

Fig 4.3.1 Graph Representing Loss and Value Loss

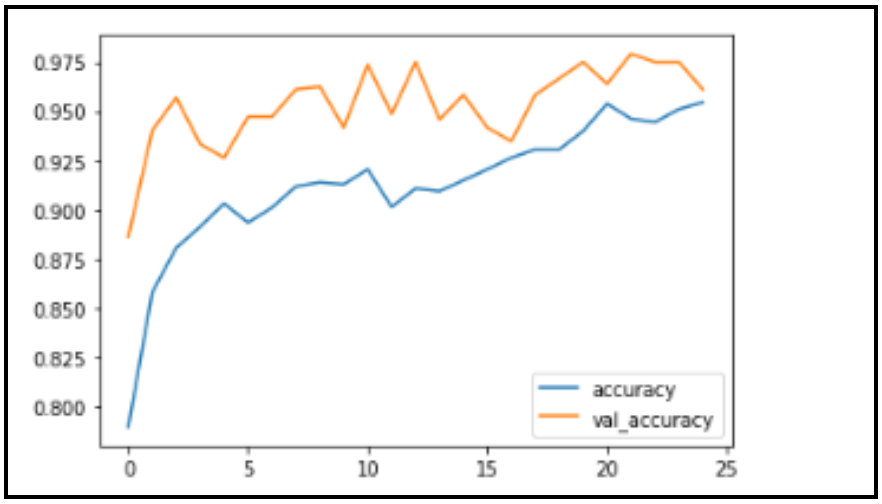

Fig 4.3.2 Graph Representing Accuracy and Value Accuracy

Prediction

```
Y_test_pred = model.predict(x_test)
Y_pred = model.predict(X_test)
print("The Accuracy of EfficientNet B7 is :",round(accuracy_score(np.argmax(y_test, axis=1), n
p.argmax(Y_test_pred, axis=1))*100,2))
```

```
The Accuracy of EfficientNet B7 is : 98.75
```

```
tta_steps = 5
predictions = []
for i in tqdm(range(tta_steps)):
    preds = model.predict_generator(train_generator.flow(x_test, batch_size=BATCH_SIZE, shuf
fle=False),
                                    steps = len(x_test)/BATCH_SIZE)
    predictions.append(preds)
    gc.collect()
Y_pred_tta = np.mean(predictions, axis=0)
```

```
100%|██████████| 5/5 [06:07<00:00, 73.52s/it]
```

Confusion Matrix

```
from sklearn.metrics import confusion_matrix
def plot_confusion_matrix(cm, classes,
```

```
                      normalize=False,
                     title='Confusion matrix',
                      cmap=plt.cm.Blues):
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
    print(cm)
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=55)
    plt.yticks(tick_marks, classes)
    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()
cm = confusion_matrix(np.argmax(y_test, axis=1), np.argmax(Y_test_pred, axis=1))
cm_plot_label =['Disease', 'Healthy']
plot_confusion_matrix(cm, cm_plot_label, title ='Confusion Matrix for Leaf Disease')
```

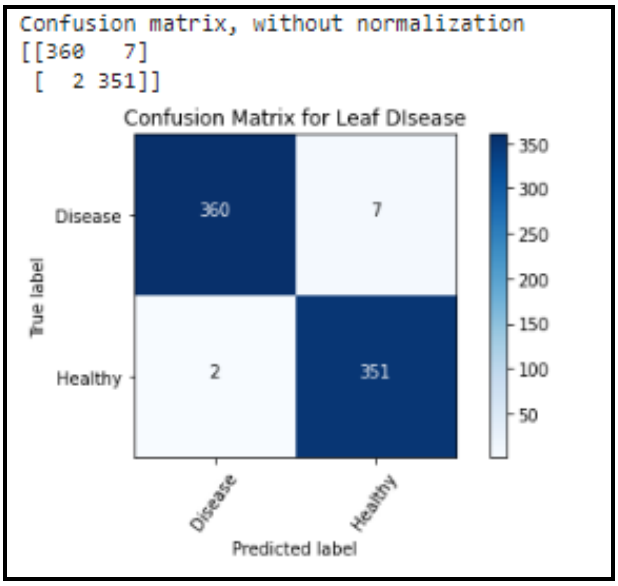

Fig 4.3.3 Confusion Matrix

Predicted Image Plot

```
i=0
prop_class=[]
mis_class=[]
for i in range(len(Y_test)):
    if(np.argmax(Y_test[i])==np.argmax(Y_pred_tta[i])):
        prop_class.append(i)
    if(len(prop_class)==8):
        break
i=0
for i in range(len(Y_test)):
    if(not np.argmax(Y_test[i])==np.argmax(Y_pred_tta[i])):
        mis_class.append(i)
    if(len(mis_class)==8):
        break
# # Display first 8 images of Plants
w=60
h=40
fig=plt.figure(figsize=(18, 10))
```

```
columns = 4
rows = 2
def Transfername(namecode):
    if namecode==0:
        return "Disease"
    else:
        return "Healthy"

for i in range(len(prop_class)):
    ax = fig.add_subplot(rows, columns, i+1)
    ax.set_title("Predicted result:"+ Transfername(np.argmax(Y_pred_tta[prop_class[i]]))
                 +"\n"+"Actual result: "+ Transfername(np.argmax(Y_test[prop_class[i]])))
    plt.imshow(X_test[prop_class[i]], interpolation='nearest')
plt.show()
```

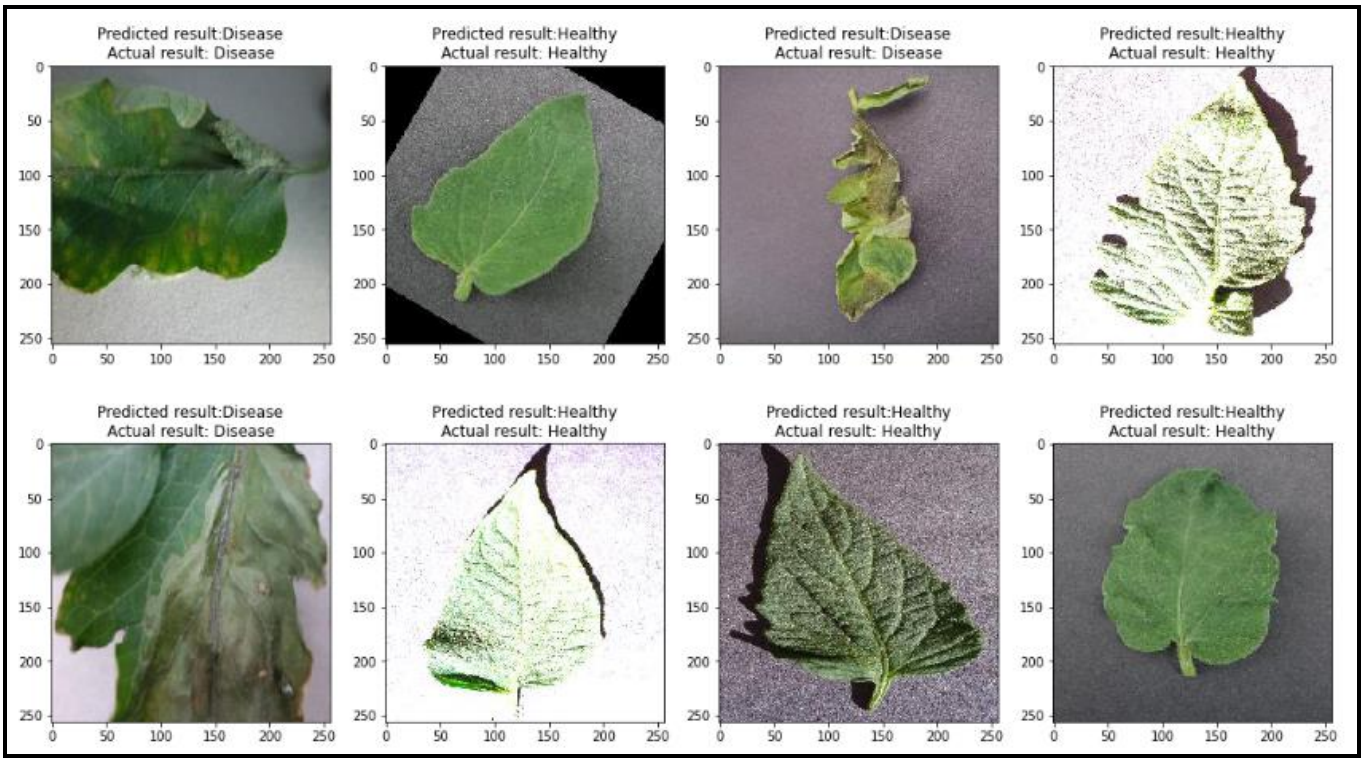

Fig 4.3.4 Predicted and Actual Result for Healthy and Disease leaf of the first 8 leafs

Chapter No. 5

CONCLUSION & FUTURE SCOPE

5.1 CONCLUSION

There are number of ways by which we can detect disease of plants and suggest remedies for them. Each has some pros as well as limitations. On one hand visual analysis is least expensive and simple method, it is not as efficient and reliable. Our goal is to help smallholder farmers grow more crops. Farmers who don't have proper knowledge about crop diseases usually feel helpless. Due to this result is less than the average yield of the crop. Therefore, the convolution neural network technique is used for classification of crop disease.

In this project, a deep learning model is trained using CNN technique for detection and classification of leaf disease. Using image processing and machine learning, we successfully identified the affected area in the plant leaf. Various features of the image are extracted with their numeric values. The algorithm used here is highly efficient and best-case time space complexities are achieved. Average accuracy of ninety plus percentage is achieved in every query image of the dataset.

5.2 FUTURE SCOPE

- To improve recognition rate of final classification process hybrid algorithms like Artificial Neural Network, Bayes classifier, Fuzzy Logic can also be used.
- Mobile application can be developed which is handy and easy to use.
- An extension of this work will focus on automatically estimating the severity of the detected disease.
- As future enhancement of the project is to develop the open multimedia (Audio/Video) about the diseases and their solution automatically once the disease is detected.
- Web based image processing techniques can be implemented. In this user can be provided with two modes with and without internet.
- In case of web base processing, remote area users can upload image in the system and whole image system techniques and classification algorithm can be implemented in the cloud itself.

REFERENCES

[1] Yashpal Sen, Chandra Shekhar Mithlesh, Dr. Vivek Baghel, A survey on crop disease detection using image processing technique for economic growth of rural area, IJARIIT(ISSN: 2454-132X), 2019.

[2] K. Elangoran, S. Nalini, 2011 "Detection and classification of leaf diseases using K-means-based segmentation and neural-networks-based classification." Inform. Technol. J., 10: 267-275. DOI: 10.3923/itj.2011.267.275. , 2019.

[3] Sandesh Raut, Karthik Ingale, "Review on leaf disease detection using Image Processing techniques."(2018).

[4] Patil, Sagar S. and Anjali Avinash Chandavale. "A Survey on Methods of Plant Disease Detection." (2015).

[5] T. Rumpf, A-K Mahlein, U sleiner, H. W. Dehne. "Texture analysis for diagnosing paddy disease." In International Conference on Electrical Engineering and Informatics, 2009. ICEEI'09., vol. 1, pp. 23-27. IEEE, 2009.

[6] Mr. Sanjay Mirchandani1, Mihir Pendse, Prathamesh Rane, Ashwini Vedula, "Plant disease detection and classification using image processing and artificial neural networks", 2018.

[7] Savita N. Ghaiwat, Parul Arora, Detection and classification of plant leaf diseases using image processing techniques: a review,Int J Recent Adv Eng Technol, 2 (3) (2014), pp. 2347-2812ISSN.

[8] Amar Kumar Deya, Manisha Sharmaa, M.R. Meshramb, "Image Processing Based Leaf Rot Disease, Detection of Betel Vine (Piper Betel.)", 2015.

[9] Jayamala k Patil, Rajkumar, "Advances in image processing for plant disease detection", 2015.

[10] S Arivazhagan, R Newlin shebiah, S Ananthi, S Vishnu varthini "Detection of unhealthy region of plant leaves and classification of plant leaf diseases using texture features."

[11] A.Singh,B.G.Subramanian,A.K.Singh,S.Sarkar, Machine Learning for High Throughput Stress Phenotyping in Plants, Trends in Plant Science, Volume 21, Issue 2, pp. 110-124, 2016.

[12] A.Camargo, J.S.Smith, An image-processing based algorithm to automatically identify plant disease visual symptoms, Biosystems Engineering ,Volume 102, Issue 1, pp. 9-21, 2009.

[13] VijaiSingh, A.K.Misra, Detection of plant leaf diseases using image segmentation and soft computing techniques, Volume 4,Issue 1, March 2017, PP. 41-49.

[14] S.Phadiar ,J.Sil, Rice Disease Identification using Pattern Recognition Techniques, Proceedings of 11th International Conference on Computer and Information Technology (ICCIT 2008), 25-27 December, pp.420 - 423, 2008.

[15] Meunkaewjinda, P. Kumsawat and K. Attakitmongcol, Grape leaf disease detection from colour imagery using hybrid intelligent system, 5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, Volume: 1, pp. 513 - 516,2008.

[16] E. Omrani, B.Khoshnevisan, S.Shamshirband, H.Saboohi, N.B.Anuar, M.H.N.M.Nasir, 'Potential of radial basis function-based support vector regression for apple disease detection', Department of Biosystem Engineering, pp.2-19, 2014.

[17] S.Bashir,N.Sharma, Remote Area Plant disease detection using Image Processing', IOSR Journal of Electronics and Communication Engineering, Volume 2, Issue 6, pp.31-34,2012.

[18] J.G.A.Barbedo,Digital image processing techniques for detecting, quantifying and classifying plant diseases, pp.1-12, Barbedo SpringerPlus ,2013.

[19] F.Qin, D.Liu, B.Sun, L.Ruan, Z.Ma, H.Wang, Identification of Alfalfa Leaf Diseases Using Image Recognition Technology, p.p.1-15, Plos Journals, December 15, 2016.

[20] Rafael C. Gonzalez and Richard E. Woods. 2006. Digital Image Processing (3rd Edition). Prentice-Hall, Inc., USA.

USER MANUAL

1. Open Google Colab and Name it to "efficientnetb7MAIN(4300Images)".
2. Select the "RunTime" from top left corner and select "Change RunTime Type" , change the "**Hardware accelerator' to "TPU" and "Runtime Shape" to "High-Ram".**
3. Link the Google Colab with Google drive where Dataset of Leaf's for Training and Validation is present.
4. Load the Dataset by inserting the filepath for each Dataset :
   - Training-containing 3600 images divided into :
     i. Healthy leaf- containing 1800 images
     ii. Disease leaf-containing 1800 images
   - Validation-containing 700 images divided into :
     iii. Healthy leaf- containing 350 images
     iv. Disease leaf-containing 350 images
   - Consisting of Total 4300 Images.
5. Build the CNN Model by using"**Efficientnetb7 model** ", and print the train features.
6. Use Classifiers "*SVM" and "Naive Bayes Classifier" to identify Accuracy,Precision,Recall and F1 score of the Validating leafs.*
7. Specify Loss Function, Optimizer and Model Summary.
8. *Perform Data Augmentation :* Data Augmentation is needed when we have less data, this boosts the accuracy of our model by augmenting the data and setting the Learning Rate Reducer to avoid Data Over fitting.
9. *Training* is done by train model using validation dataset to validate each steps and Evaluation is done by plotting the graph for Training and Validation Accuracy and Training and Validation Loss.
10. Prediction is done based on the accuracy score of the model EfficientNet B7.
11. Execute Confusion Matrix: It helps us evaluate how our model performed, where it went wrong and offers us guidance to correct our path and show the Confusion matrix for the Leaf Disease for True and Predicted Label.
12. Execute the Predicted Image Plot for first 8 Images of leafs with the Actual and Predicted Results.