VELS

INSTITUTE OF SCIENCE TECHNOLOGY & ADVANCED STUDIES (VISTAS)

(Deemed to be University under section 3 of UGC Act, 1956)

NAAC ACCREDITED WITH 'A' GRADE

Crop Health Analysis using Deep Learning and Artificial Neural Network

KAUSHIK K **19602904**

VIKRAM RAJA **19602210**

VELS INSTITUTE OF SCIENCE, TECHNOLOGY AND ADVANCED STUDIES (VISTAS)

A FINAL YEAR PROJECT REPORT

Submitted to the

SCHOOL OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

In Partial fulfillment of the requirements

for the award of the degree of

BACHELOR OF ENGINEERING IN

COMPUTER SCIENCE AND ENGINEERING

VELS INSTITUTE OF SCIENCE, TECHNOLOGY AND ADVANCED STUDIES (VISTAS)

CHENNAI – 600 117

NOVEMBER 2022

BONAFIDE CERTIFICATE

Certified that this project **"Crop Health Analysis using Deep Learning and Artificial Neural Network"** is the bonafide work of **Kaushik K (19602904) and Vikram Raja (19602210)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported here does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr.R.Anandan M.E., Ph.D,

Professor
Head of the department

Dept. Computer Science & Engineering

VISTAS

Chennai – 600 117

SIGNATURE

Ms.Packilatha M.E., Ph.D,

Assistant Professor
Supervisor

Dept. Computer Science & Engineering

VISTAS

Chennai – 600 117

Submitted for the project work and viva - voice examination held on

…… at Vels Institute of Science, Technology

and Advanced Studies (VISTAS), Chennai – 600 117.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

It is certainly a great delight and proud privilege to acknowledge the help and support I received from the positive minds around me in making this venture a successful one. The infrastructural support with all kinds of lab facilities have been a motivating factor in this completion of project work, all because of **FOUNDER & CHAIRMAN, Shri. Dr. ISHARI K. GANESH.,**with great pleasure that I take this opportunity to thank him. I am grateful to our **HEAD OF THE DEPARTMENT, Dr.ANANDAN M.E., (Ph.D.),** for her continuous support for the project, from initial advice and contacts in the early stages of conceptual inception, and through on-going advice and encouragement to this day. I extend our warmest thanks to my **INTERNAL GUIDE, Ms.PACKIALATHA M.E., Ph.D.,** who has patiently guided and provided us with invaluable advice and help. I would like to express my greatest gratitude to our staff members, librarian and non-teaching staff members of Computer Centre, who have helped and supported me throughout. Last but not the Least, I wish to thank my parents and my friends for their individual support and interest who has inspired me and encouraged me to go in my own way, without them I would be unable to complete this project.

KAUSHIK K - 19602904
VIKRAM RAJA - 19602210

TABLE OF CONTENTS

CHAPTER 1

ABSTRACT

Economic growth of a country depends on its agricultural productivity. Identification of the plant diseases is one of the key aspects for preventing the losses in the agricultural production and improving the quality of the product. Traditional methods are reliable but require a human resource for visually observing the plant leaf patterns and diagnose the disease. This consumes more time and tedious work for labourers. In big farmlands, early-stage detection of plant disease by using automated techniques will reduce the loss in productivity. In this project/paper, we propose a vision based automatic detection of plant disease detection using Image Processing Technique. Image processing algorithms are developed to detect the plant infection or disease by identifying the color feature of the leaf area. K mean algorithm is used for color segmentation, gaussian functions for filtration and GLCM is used for diseases classification. Vision based plant infection showed efficient result and promising performance. Adding on this we have implemented several other features which are induced using the image processing technique. Once the image processing is done the model would be capable of identifying the plant species along with its growth pattern among a cluster of crops. It doesn't stop there if a crop had been diagnosed with a particular disease, our model would suggest a set of curative procedures to stop further spread and enriching the farmers with a healthy and a plentiful yield.

Aim/Scope

The proposed methodology is used for the precise detection of disease in crops. Which can provide controlled fertilization to farmers. Accurate identification of disease also helps farmers to identify the infection and do relatively controlled fertilization to avoid any future crop failures. Our aim is to design an AI-Based disease detection system that detects the type of disease present in the leaves of plants such as apple, cherry, blueberry, corn, strawberry etc., by clicking the images of various leaves through the camera and determine the health status of the crop through classification done by Deep Convolutional neural networks (D-CNN) and transfer learning techniques used to detect and classify the disease. If diagnosed with a disease/infection, propose a curative procedure to further prevent the spread.

Objectives

There are three objectives of the proposed methodology:

i To develop a prototype for a plant disease detection system.

ii To apply image processing techniques to identify the disease pattern.

iii Use machine learning algorithms to predict disease.

iv Make it user-friendly for the users and easily available.

Organization of the project:

The report is divided into eight chapters.

Chapter 1: gives a brief introduction of the project covered and the aim of the project.

Chapter 2: aims at the literature survey of the project consisting of the basic idea of the project, and how we got the idea to make this project, with the help of websites, ieee papers etc.

Chapter 3: covers introduction of Image processing and steps in Image processing

Chapter 4: covers the block and UML diagram of the entire module, flowchart.

Chapter 5: covers the metadata of the dataset

Chapter 6: deals with the results

Chapter 7: advantages and applications

Chapter 8: covers all the references used to complete the project.

CHAPTER 2

LITERATURE SURVEY

Title: A review of advanced techniques for detecting plant diseases.

Year: 2010

Diseases in plants cause major production and economic losses in agricultural industry worldwide. Monitoring of health and detection of diseases in plants and trees is critical for sustainable agriculture. To the best of our knowledge, there is no sensor commercially available for real-time assessment of health conditions in trees. Currently, scouting is most widely used mechanism for monitoring stress in trees, which is an expensive, labor-intensive, and time-consuming process. Molecular techniques such as polymerase chain reaction are used for the identification of plant diseases that require detailed sampling and processing procedure. Early information on crop health and disease detection can facilitate the control of diseases through proper management strategies such as vector control through pesticide applications, fungicide applications, and disease-specific chemical applications; and can improve productivity. The present review recognizes the need for developing a rapid, cost-effective, and reliable health-monitoring sensor that would facilitate advancements in agriculture. It describes the currently used technologies that can be used for developing a ground-based sensor system to assist in monitoring health and diseases in plants under field conditions. These technologies include spectroscopic and imaging-based, and volatile profiling-based plant disease detection methods. The paper compares the benefits and limitations of these potential methods.

Title: A Capacitive to digital Converter with Automatic Range Adaptation

Year: 2016

Capacitive sensors have profoundly found their way in everyday life. Devices and instrumentation ranging from specialty equipment to smart phones all employ in one way or the other a capacitive sensor and its associated readout circuit, making the latter ubiquitous. We present a capacitive readout system that automatically adapts its range to the unknown measured

instrumentation compatibility. The proposed system achieves a constant resolution for a range of input capacitance up to 690 pF.

Title: Near-infrared spectroscopy for the prediction of disease ratings for Fiji leaf gall in sugarcane clones.

Year: 2009

This paper demonstrates how inferential measurements or indirect methods using near-infrared (NIR) methodology and chemo metrics can be used to predict sugarcane clonal performance. Fiji leaf gall resistance is used in this study as an example. Fiji leaf gall is one of Australia's most serious sugarcane diseases, representing a significant problem in almost half of the total area under production. Traditional rating of sugarcane clones for resistance/susceptibility is difficult and expensive because of the nature of field-based methods and variable infection levels of the trials. Thus, the aim of this work was to investigate the potential of NIR spectroscopy as an alternative means to rate clones from direct measurement of sugarcane leaf spectra and to examine its ability to successfully predict traditional resistance ratings using a calibration model based on a chemo metrics method such as partial least squares (PLS). A scanning electron microscopy (SEM) study of the leaf substrate was undertaken to elucidate the nature of the NIR sample site. In addition, an NIR study of freeze-dried sugarcane leaf samples resolved the heavily overlapping O-H bands present in the NIR spectrum due to water/cellulose interaction. A significant decrease in the spectral intensity between 5205 and 5393 cm (-1) was observed and a similar decrease was noted in the OH stretching overtone (7114 cm(-1)) with an accompanying shift to lower wave numbers. PLS modeling based on traditional ratings as the dependent variable and the corresponding NIR spectra showed satisfactory results with standard error of validation (SEV) and standard error of prediction (SEP) values being 0.98 (R(2) = 0.97) and 1.20 (R(2) = 0.88), respectively. This methodology has now been recommended for more extensive field trials.

Title: An Image-Based Diagnostic Expert System for Corn Diseases

Year: 2010

The annual worldwide yield losses due to pests are estimated to be billions of dollars. Integrated pest management (IPM) is one of the most important components of crop production in most agricultural areas of the world, and the effectiveness of crop protection depends on accurate and timely diagnosis of phytosanitary problems. Accurately identifying and treatment depends on the method which used in disease and insect pests' diagnosis. Identifying plant diseases is usually difficult and requires a plant pathologist or well-trained technician to accurately describe the case. Moreover, quite a few diseases have similar symptoms making it difficult for non-experts to distinguish disease correctly. Another method of diagnosis depends on comparison of the concerned case with similar ones through one image or more of the symptoms and helps enormously in overcoming difficulties of non-experts. The old adage 'a picture is worth a thousand words' is crucially relevant. Considering the user's capability to deal and interact with the expert system easily and clearly, a web-based diagnostic expert-system shell based on production rules (i.e., IF < effects > THEN < causes >) and frames with a color image database was developed and applied to corn disease diagnosis as a case study. The expert- system shell was made on a 32-bit multimedia desktop microcomputer. The knowledge base had frames, production rules and synonym words as the result of interview and arrangement. It was desired that 80% of total frames used visual color image data to explain the meaning of observations and conclusions. Visual color image displays with the phrases of questions and answers from the expert system, enables users to identify any disease, makes the right decision, and chooses the right treatment. This may increase their level of understanding of corn disease diagnosis. The expert system can be applied to diagnosis of other plant pests or diseases by easy changes to the knowledge base.

Title: Applying image processing technique to detect plant diseases.

Year: 2012

The present work proposes a methodology for detecting plant diseases early and accurately, using diverse image processing techniques and artificial neural network (ANN). Farmers experience great difficulties in changing from one disease control policy to another. Relying on pure naked eye observation to detect and classify diseases can be expensive various plant diseasespose a great threat to the agricultural sector by reducing the life of the plants. the present work is aimed to develop a simple disease detection system for plant diseases. The work begins with capturing the images. Filtered and segmented using Gabor filter. Then, texture and color features are extracted from the result of segmentation and Artificial neural network (ANN) is then trained by choosing the feature values that could distinguish the healthy and diseased samples appropriately. Experimental results showed that classification performance by ANN taking feature set is better with an accuracy of 91%.

CHAPTER 3

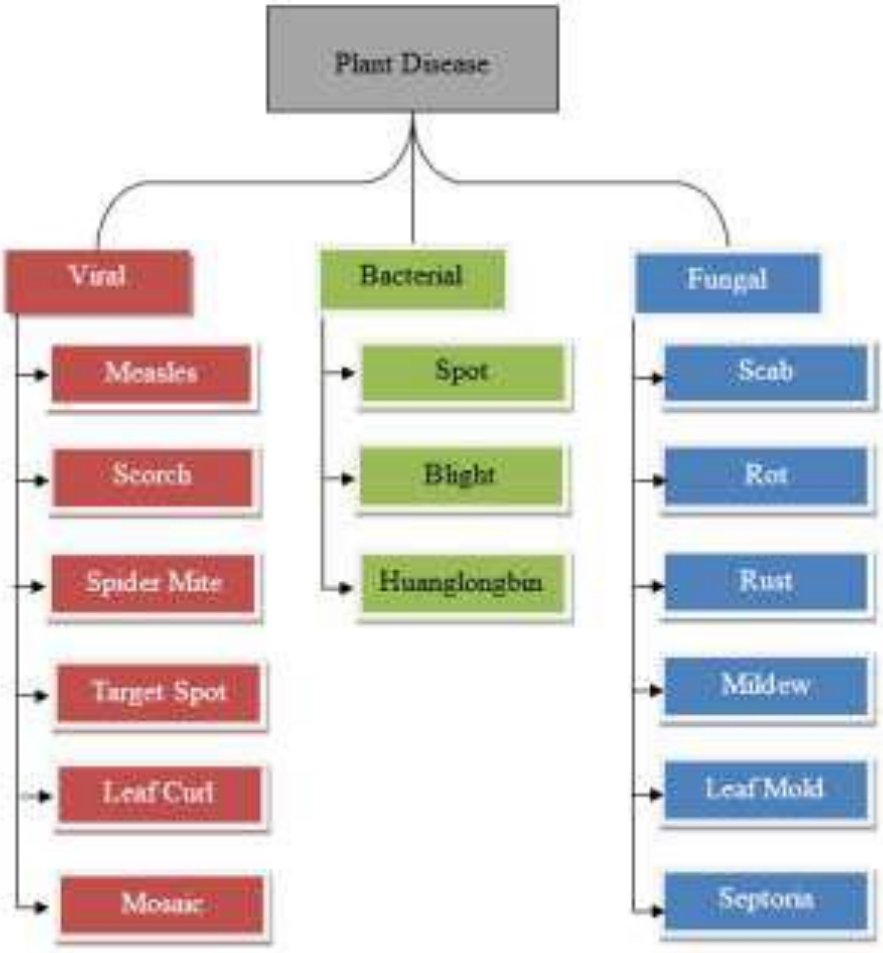
Introduction about different types of plant diseases

The project is based on automatic plant disease detection. It is necessary to know about different types of plant diseases. Research about different types of plant disease is carried out. We found the major diseases that affect plants are Bacterial Spot, Bacterial Blight, Ralstonia solanacearum, Downy Mildew, Agular Leaf Spot, Anthracnose, Alternaria Alterneta, CercosporinLeaf Spot, Botrytis etc.

Each disease has a particular symptom which appears on the leaf. The plant diseases are caused by bacteria, viruses, prolonged wetness, seed and plant debris, over watering ,humidity and overhead irrigation, warm and wet environments, by various pathogens and weeds etc.

Bacterial Spot is caused by warm and wet environments. It looks like small, dark, raised spots. Bacterial Blight is mainly caused due to cool and wet weather. It appears on the leaf as large. Yellow spots that eventually turns brown. Ralston solanacearum also called as Bacterial Wilt is caused by contaminated soils or weeds. Its looks like wilted leaves on plants during the daytime, eventually turning yellow and remaining wilted. Downey Mildew caused due to prolonged wetness. It appears as white mildew typically on underside of leaves. A regular leaf spot is caused by remaining of seed and plant debris in the surrounding of plant. Holes in the leaves are appeared when plant is affected by this disease. Anthracnose is caused by fungus. The fungal disease overwinters in and on seeds, soil and garden debris. Cool wet weather promotes its development. Infected plants develop dark, water-soaked lesions on stems, leaves or fruit. The centers of these lesions often become covered with pink, gelatinous masses of spores especially during moist, warm weather. Alternaria alternate is a fungus which has been recorded causing leaf spot and other diseases on over 380 host species of plant. It is an opportunistic pathogen on numerous hosts causing leaf spots, rots and blights on many plant parts. Cercosporin leaf spot can be caused by many different Cercosporin fungal pathogen species depending on the plant type infected. This disease causes spots on foliage. The circular blemishes typically have an average diameter of 3 millimeters that is between brown and reddish-purple in color, bordering a gray center. However, if spots have just developed, the gray center will not yet be visible. When this fungus experiences favorable conditions the growth it truly viible.

This basic learning of different plant diseases and their symptoms that appear on leaves has been of great use as it helped understand how professionals approach the task of recognizing the leave diseases. But the approach through naked eye for detecting plant diseases may give wrong prediction of the disease in the plants. Hence it requires laboratory tests to be carried out by professionals. But laboratories are not available everywhere especially in rural areas. For this purpose, automatic plant disease using digital image processing techniques by using images of disease affected plant leaf is helpful to detect the disease in early stage. The features like color, shape, texture on affected plant leaf helps for classifying and detecting the diseases.



METHODOLOGY

Proposed System:

The basic of the project is to detect the plant diseases and suggest solutions to recover from the leaf diseases. We have planned to design our project with digital image processing so that a person with lesser expertise in the software field should also be able to use it easily and is such our user-friendly UI. In our system we are providing a solution to recover from the leaf diseases and show the affected part of the leaf by image processing technique. The existing system only identify the type of diseases which affects the leaf. We will provide a result within seconds and guided you throughout the project. In this we have trained data sets and training data sets. Trained data sets which contain a number of different infected leaves from a plethora of pants. Samples of images are collected that comprised of different leaf diseases like Alternaria Alternata, Anthracnose, Bacterial Blight, Cercospora leaf spot and Healthy Leaves. The main objective is to identify the plant diseases using image processing. The remedy of disease is shown by the system after the identification of the diseases.

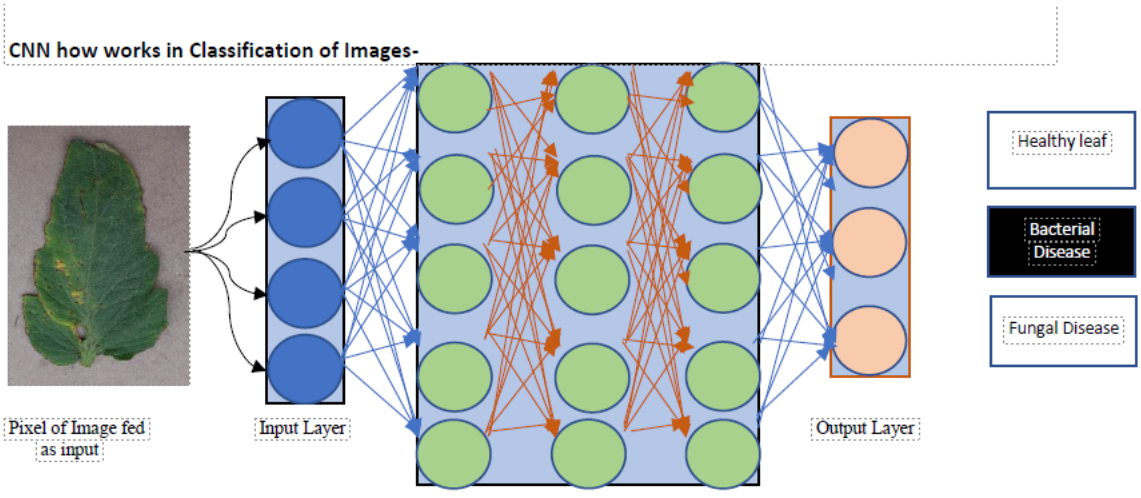


Fig.2 Model of CNN Network

CNN and how it works in the Classification of Images

1. Input layer-In the above figure Pixel of image fed as input then the image given in input layer where it accepts the pixel of the image as input in the form of array.

2. Hidden Layer -Hidden layers carry out feature extraction by performing certain calculation and manipulation.

- **Convolution layer-** This layer uses a matrix filter and performs convolution operation to detect patterns in the image.

- **ReLU**- ReLU activation function is applied to the convolution layer to get a rectified feature map of image

- **Pooling**- Pooling layer also uses multiple filters to detect edges, corners, features etc.

3. Output Layer-Finally there is fully connected layer that identifies the object in the image.

The proposed system takes input image for testing after that pre-processing of image are carried out. Input image are converted to an array for compression of image according to image size given. On other side segregated database are created, in this also a pre-processing of data is done. Image form the segregated database are classified using CNN classification. Input image after converted to an array after compression, CNN based classification are applied to the trained dataset (segregated database) and testing dataset (input image) and finally defect of the leaf are display. If defect is found display disease and Remedy otherwise if defect not found, then leaf is Healthy.

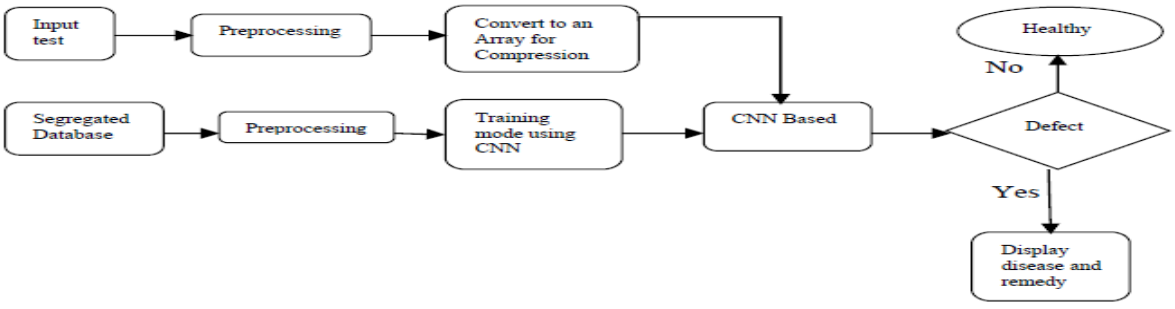


Fig. 3 Flow chart of leaf disease detection

Introduction to Image Processing

Signal processing is a discipline in electrical engineering and in mathematics that deals with analysis and processing of analog and digital signals, and deals with storing, filtering, and other operations on signals. These signals include transmission signals, sound or voice signals, image signals, and other signals etc.

Out of all these signals, the field that deals with the type of signals for which the input is an image, and the output is also an image is done in image processing. As the name suggests, it deals with the processing on images.

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image. Nowadays, image processing is among rapidly growing technologies. It forms core research area within engineering and computer science disciplines too.

It can be further divided into analog image processing and digital image processing.

1. Analog image processing

   Analog image processing is done on analog signals. It includes processing on two dimensional analog signals. In this type of processing, the images are manipulated by electrical means by varying the electrical signal. The common example include is the television image.

   Digital image processing has dominated over analog image processing with the passage of time due its wider range of applications.

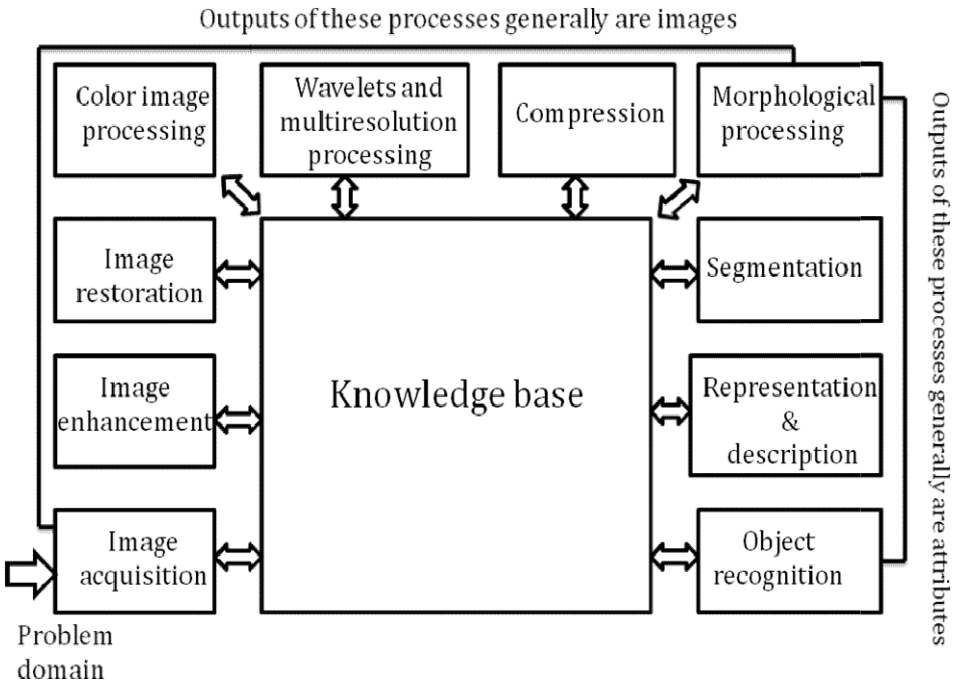
2. Digital image processing

   Digital image processing, the manipulation of images by computer, is relatively recent development in terms of man’s ancient fascination with visual stimuli. In its short history, it has

been applied to practically every type of images with varying degree of success. The inherent subjective appeal of pictorial displays attracts perhaps a disproportionate amount of attention from the scientists and also from the layman. Digital image processing like other glamour fields, suffers from myths, mis-connect ions, mis-understandings and mis-information. It is vast umbrella under which fall diverse aspect of optics, electronics, mathematics, photography graphics and computer technology.

Several factor combine to indicate a lively future for digital image processing. A major factor is the declining cost of computer equipment. Several new technological trends promise to further promote digital image processing. These include parallel processing mode practical by low cost microprocessors, and the use of charge coupled devices (CCDs) for digitizing, storage during processing and display and large low cost of image storage arrays.

Fundamental steps in Image Processing



Fundamental steps in Image Processing

Image Acquisition:

Image Acquisition is to acquire a digital image. To do so requires an image sensor and the capability to digitize the signal produced by the sensor. The sensor could be monochrome or

color TV camera that produces an entire image of the problem domain eve ry 1/30 sec. the image sensor could also be line scan camera that produces a single image line at a time. In this case, the objects motion past the line.

Scanner produces a two-dimensional image. If the output of the camera or other imaging sensor is not in digital form, an analog to digital converter digitizes it. The nature of the sensor and the image it produces are determined by the application.

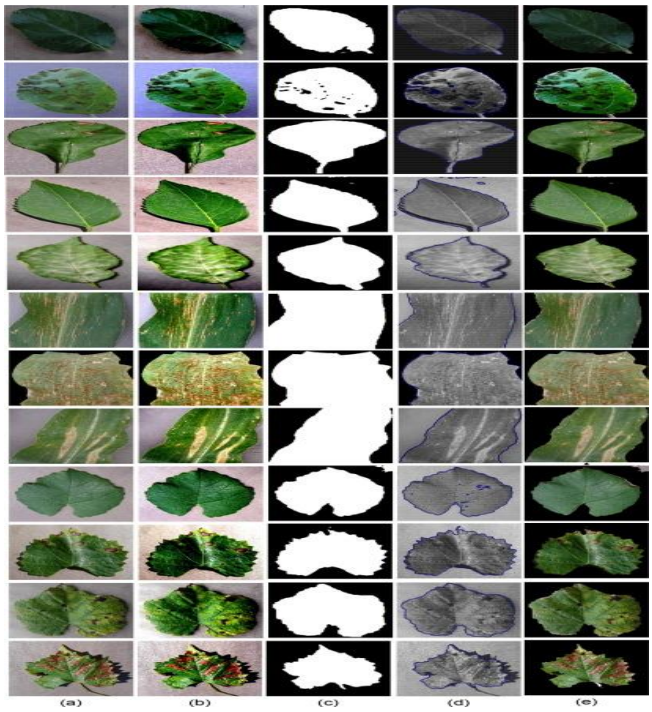
Image Enhancement:

Image enhancement is among the simplest and most appealing areas of digital image processing. Basically, the idea behind enhancement techniques is to bring out detail that is obscured, or simply to highlight certain features of interesting an image.

It is important to keep in mind that enhancement is a very subjective area of image processing.

Image restoration:

Image restoration is an area that also deals with improving the appearance of an image. However, unlike enhancement, which is subjective, image restoration is objective, in the sense that restoration techniques tend to be based on mathematical or probabilistic models of image degradation.



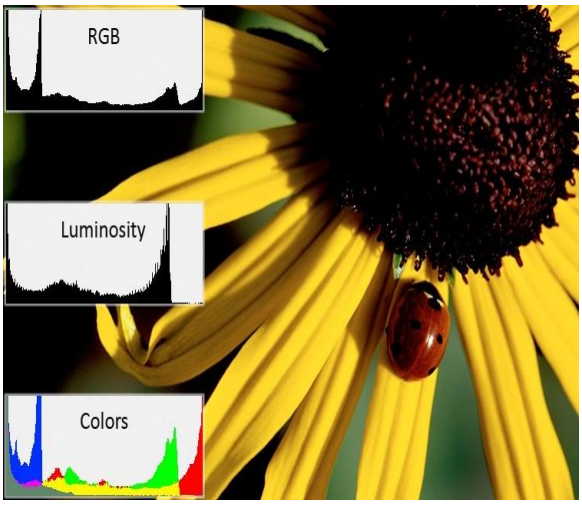
Enhancement, on the other hand, is based on human subjective preferences regarding what constitutes a "good" enhancement result. For example, contrast stretching is considered an enhancement technique because it is based primarily on the pleasing aspects it might present to the viewer, whereas removal of image blur by applying a deblurring function is considered a restoration technique.

Color image processing:

The use of color in image processing is motivated by two principal factors. First, color is a powerful descriptor that often simplifies object identification and extraction from a scene. Second, humans can discern thousands of color shades and intensities, compared to about only two dozen shades of gray. This second factor is particularly important in manual image analysis.

WagesWavelets and multi resolution processing:

Wavelets are the formation for representing images in various degrees of resolution. Although the Fourier transform has been the mainstay of transform based image processing since the late1950's, a more recent transformation, called the wavelet transform, and is now making it even easier to compress, transmit, and analyse many images. Unlike the Fourier transform, whose basis functions are sinusoids, wavelet transforms are based on small values, called Wavelets, of varying frequency and limited duration.



Wavelets were first shown to be the foundation of a powerful new approach to signal processing and analysis called **Multiresolution** theory. Multiresolution theory incorporates and unifies techniques from a variety of disciplines, including sub band coding from signal processing, quadrature mirror filtering from digital speech recognition, and pyramidal image processing.

Compression:

Compression, as the name implies, deals with techniques for reducing the storage required saving an image, or the bandwidth required for transmitting it. Although storage technology has improved significantly over the past decade, the same cannot be said for transmission capacity. This is true particularly in uses of the Internet, which are characterized by significant pictorial content. Image compression is familiar to most users of computers in the form of image file extensions, such as the jpg file extension used in the JPEG (Joint Photographic Experts Group) image compression standard.

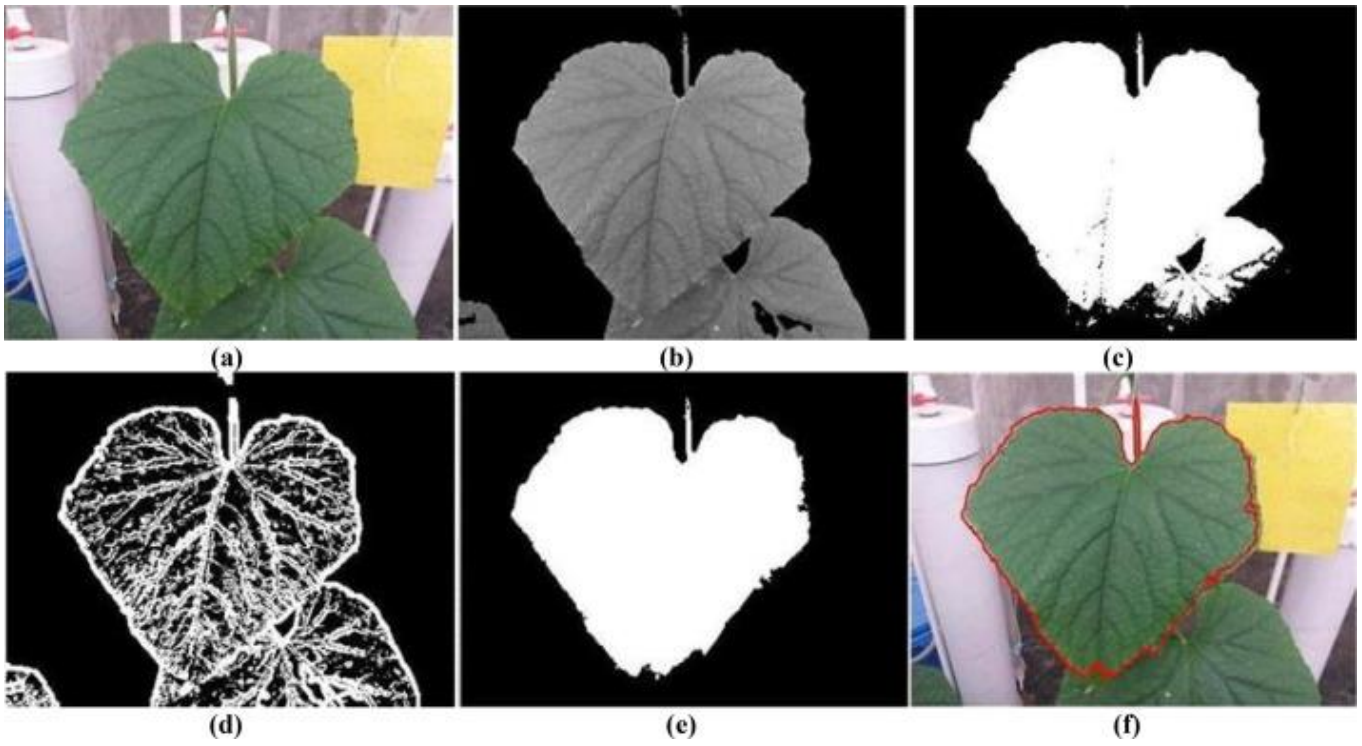
Morphological processing:

Morphological processing deals with tools for extracting image components that are useful in the representation and description of shape. The language of mathematical morphology is set theory. As such, morphology offers a unified and powerful approach to numerous image processing problems. Sets in mathematical morphology represent objects in an image. For example, the set of all black pixels in a binary image is a complete morphological description of the image.

In binary images, the sets in question are members of the 2-D integer space Z^2, where each element of a set is a 2-D vector whose coordinates are the (x,y) coordinates of a black(or white) pixel in the image. Gray-scale digital images can be represented as sets whose components are in Z^3. In this case, two components of each element of the set refer to the coordinates of a pixel, and the third corresponds to its discrete gray-level value.

Segmentation:

Segmentation procedures partition an image into its constituent parts or objects. In general, autonomous segmentation is one of the most difficult tasks in digital image processing. A rugged segmentation procedure brings the process a long way toward successful solution of imaging problems that require objects to be identified individually.



Segmented image

On the other hand, weak or erratic segmentation algorithms almost always guarantee eventual failure. In general, the more accurate the segmentation, the more likely recognition is to succeed.

Representation and description:

Representation and description almost always follow the output of a segmentation stage, which usually is raw pixel data, constituting either the boundary of a region (i.e., the set ofpixels separating one image region from another) or all the points in the region itself. In either case, converting the data to a form suitable for computer processing is necessary. The first decision that must be made is whether the data should be represented as a boundary or as a complete region. Boundary representation is appropriate when the focus is on external shape characteristics, such as corners and inflections.

Regional representation is appropriate when the focus is on internal properties, such as texture or skeletal shape. In some applications, these representations complement each other. Choosing a representation is only part of the solution for transforming raw data into a form suitable for subsequent computer processing. A method must also be specified for describing the data so that features of interest are highlighted. Description, also called feature selection, deals with extracting attributes that result in some quantitative information of interest or are basic for differentiating one class of objects from another.

Object recognition:

The last stage involves recognition and interpretation.

Recognition is the process that assigns a label to an object based on the information provided by its descriptors. Interpretation involves assigning meaning to an ensemble of recognized objects.

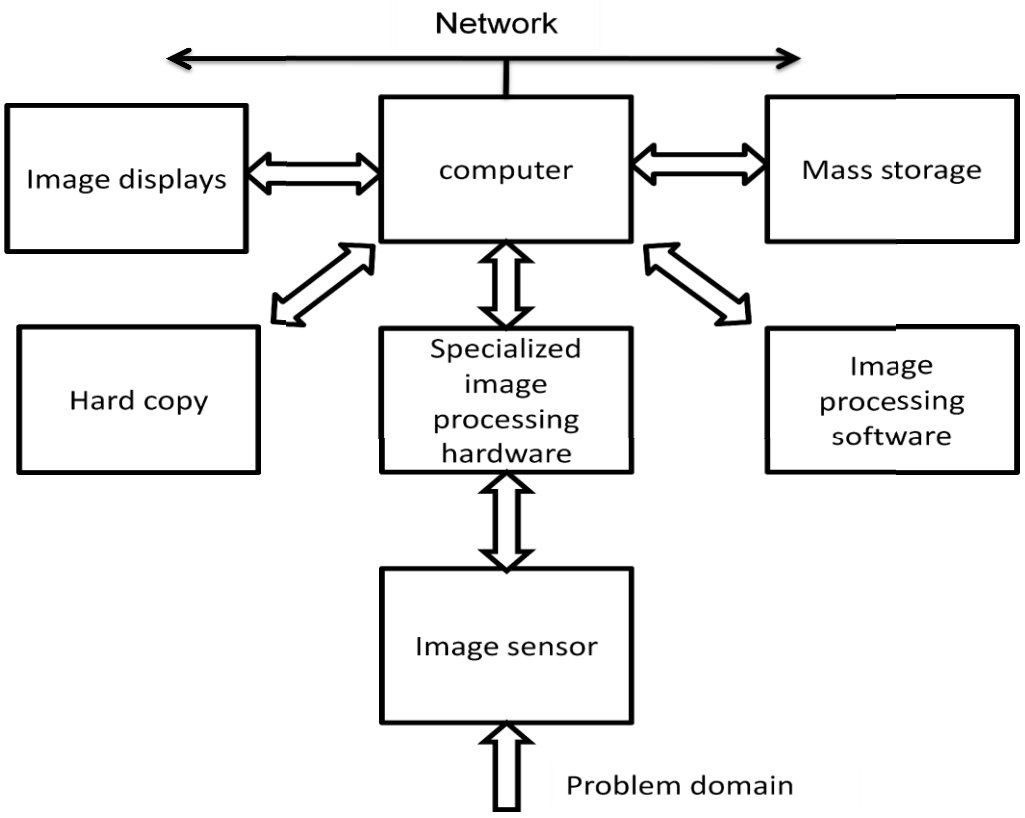
Knowledgebase:

Knowledge about a problem domain is coded into image processing system in the formof a knowledge database. This knowledge may be as simple as detailing regions of an image when the information of interests is known to be located, thus limiting the search that has to be conducted in seeking that information. The knowledge base also can be quite complex, such as an interrelated to list of all major possible defects in a materials inspection problem or an image data base containing high resolution satellite images of a region in connection with change deletion application.

Components of an Image Processing system

wledge base.

As recently as the mid-1980s, numerous models of image processing systems being sold throughout the world were rather substantial peripheral devices that attached to equally substantial host computers. Late in the 1980s and early in the 1990s, the market shifted to image processing hardware in the form of single boards designed to be compatible with industry standard buses and to fit into engineering workstation cabinets and personal computers. In addition to lowering costs, this market shift also served as a catalyst for a significant number of new companies whose specialty is the development of software written specifically for image processing.

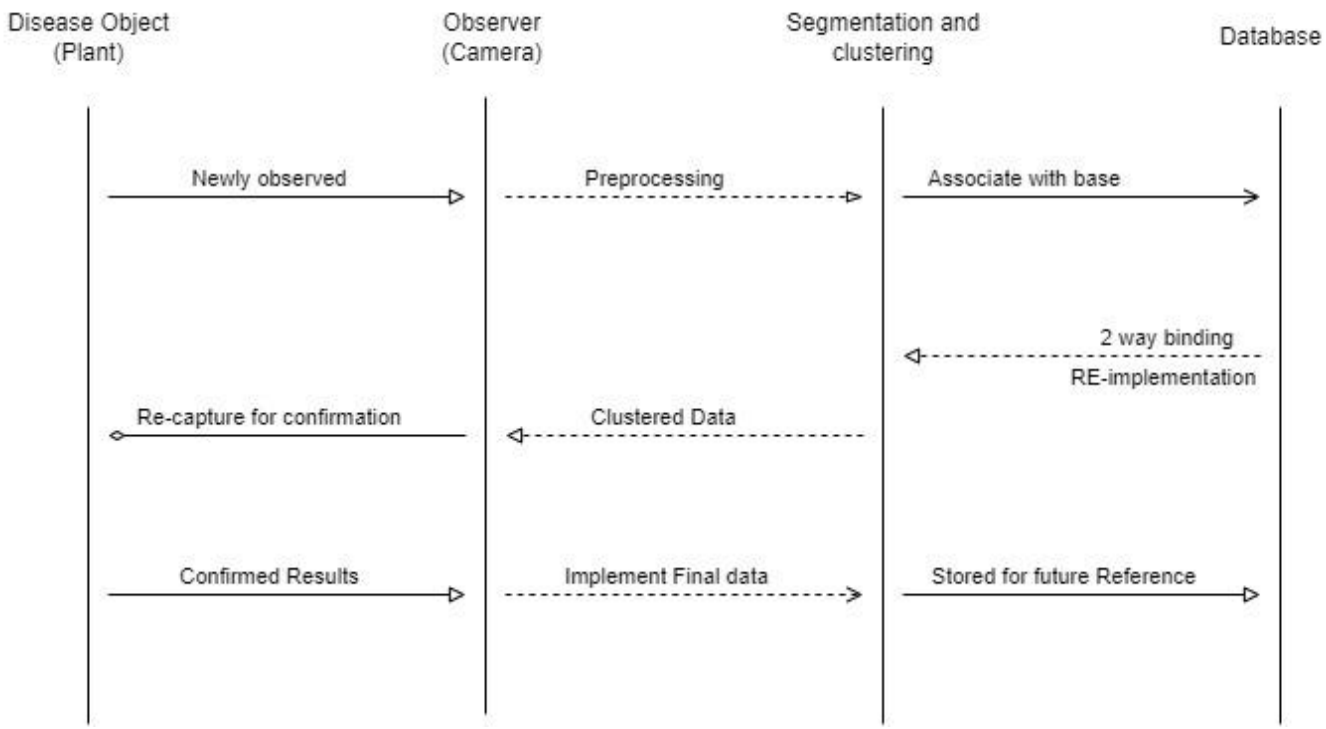


CHAPTER 4

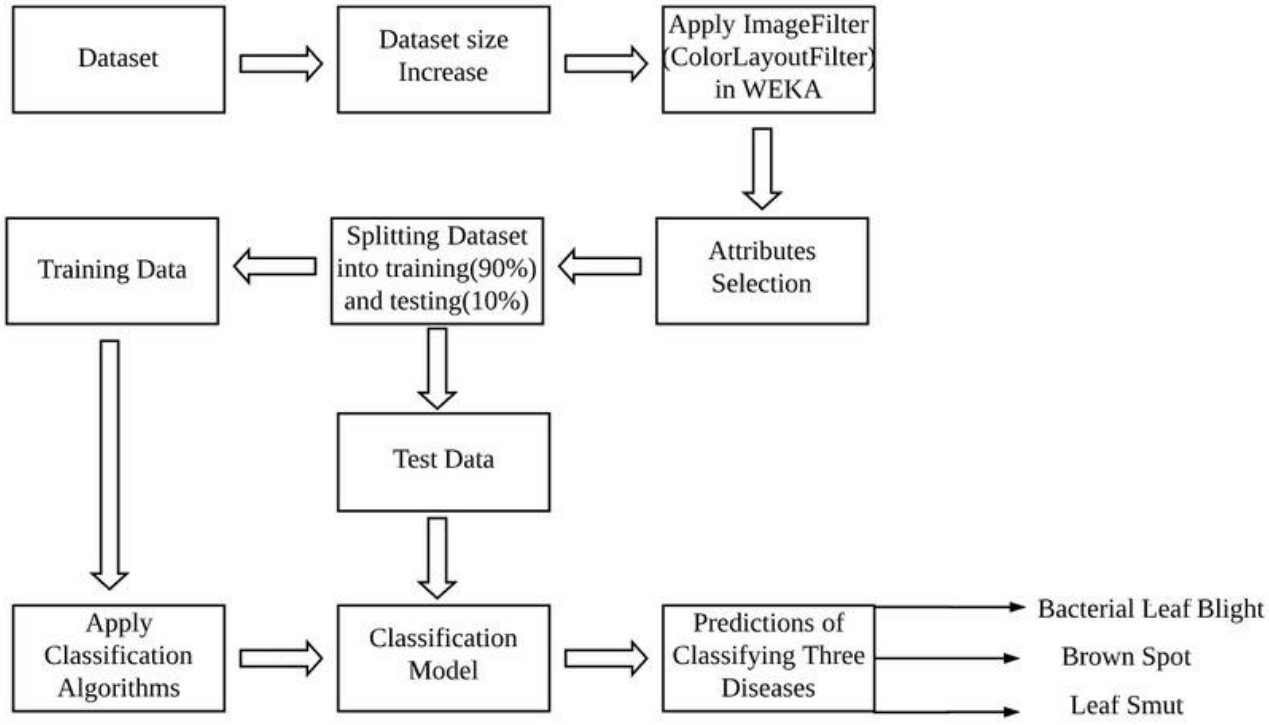
PROJECT DESIGN FLOW

UML diagram

Image processing technique that can be implemented to detect the diseases in plants is proposed in project. The system isolates the lesions (or spots) that can appear at various parts of a plant like the leaves, or the fruit. The diagnosis is based on the number of spots, their area and their color features. These features are compared with predetermined limits in order to select the matching disease. A preliminary approach has been presented. This earlier approach focused on the accurate estimation of the spot features. The application has been tested here for vineyard diseases based on photographs with grape leaves. A success rate higher than 90% has been achieved in the disease recognition process. The quantified features used for the recognition of the supported plant diseases can be announced to the user making the system easily extendible to new diseases.



Architecture/Block diagram:

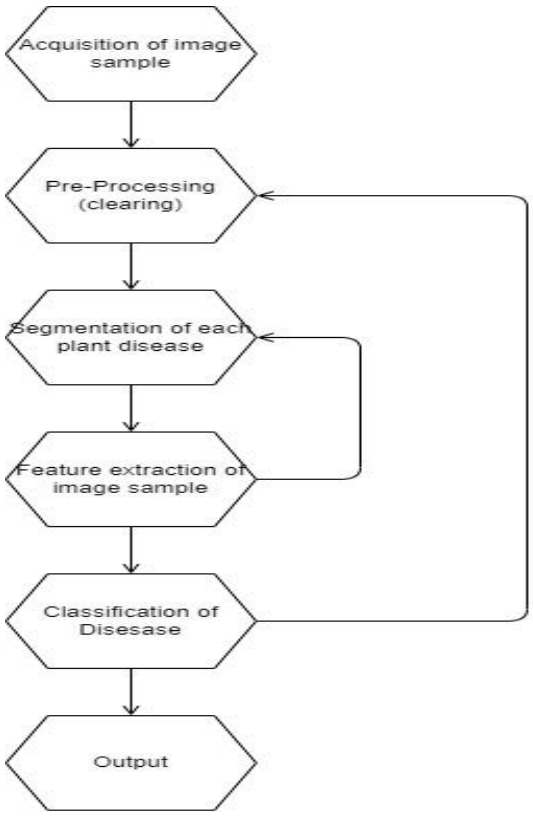


As the user adds on to the dataset by clicking pictures in real-time to check for the health status of the crop, the image data adds upon the existing dataset. Enter our model, applies filters and selectively classifies the attributes.

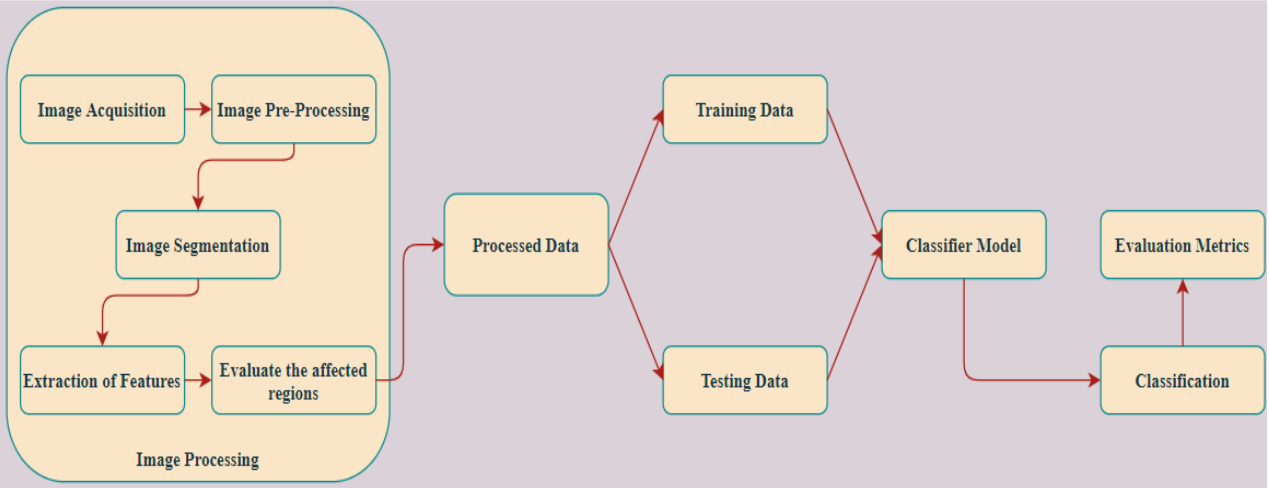
The data is then split into training and test datasets, where training dataset is used to train the model by applying classification algorithms and comes up with a classification model which is cross referenced with the test data to check the accuracy of the trained model's prediction. The prediction is done with certain disease based terms Bacterial leaf blight, Brown spot, Leaf smut etc.

Flow Chart

Flowchart of the flow of model



Flow chart describes the flow of processes done in the implementation of the project. An image of disease affected leaf is acquired first, using digital camera. This image is fed asinput for image pre-processing. In image pre-processing, image enhancement is done. The enhanced image is segmented using **pytorch extended by torchvision and python image library**(PIL) . Various features of segmented cluster image are extracted using GLCM. At the end, SVM technique is used for classification anddisease detection.



Flow Chart of the Machine Learning Methodology

Implementation Steps of the Model

1. Import image:

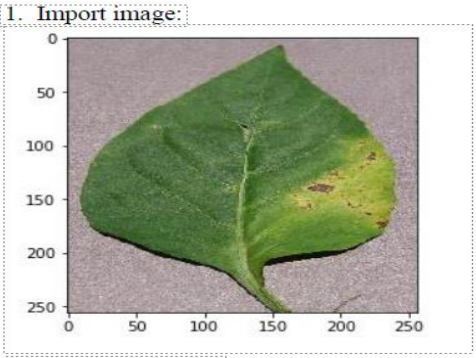


Figure 1: Input Image

2. Resize image:

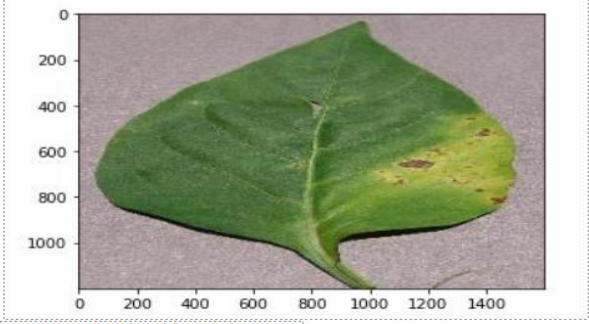


Figure 2: Resized input image

3. Blurring the image

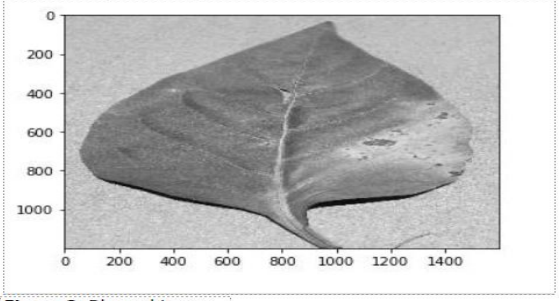


Figure 3: Blurred Image

4. Otsu segmentation to identify the infected region

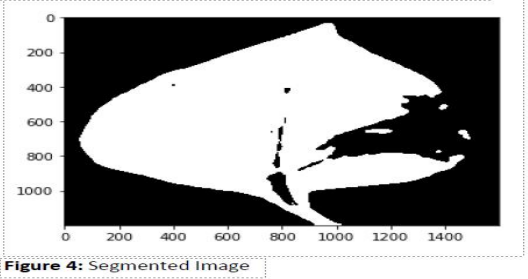


Figure 4: Segmented Image

5. Create a mask image for background subtraction using leaf contour

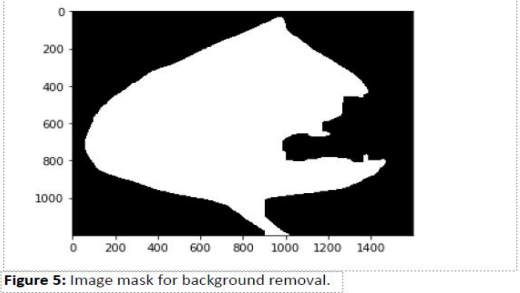


Figure 5: Image mask for background removal.

6. Performing masking operation on original image:

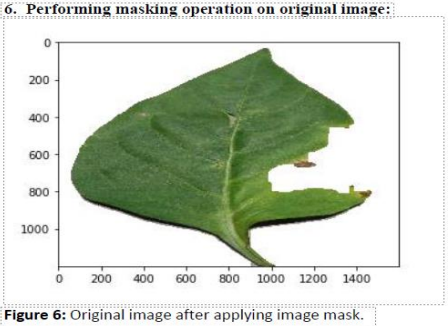


Figure 6: Original image after applying image mask.

A series of steps need to be carefully followed for the process need to be followed in a disciplined manner:

Step-1: Image Acquisition for dataset creation: This step involves exploring various data sources from where data can be extracted for training the model and further how the test image input is to be provided.

Step-2: Image Pre-processing and background removal: This is most important phas, as it involves the quality assurance of the data. In the image pre-processing phase image is processed to desired color format, resized to desired size and images are denoised.

Step-3: Image Segmentation to obtain infected region: Region of interest that is the infected part of the leaf is identified. This is again one of the most crucial step, as entire analysis is dependent on the infected refion identified by the process of segmentation.

Step-4: Extraction of Features from images: On the basis of obtained region of interest which is the infected part of the leaf various image features like standard deviation, mean of red, blue and green channels, the entropy of image is extracted.

Step-5: Evaluate and identification of the affected region: By comparing the extracted region of interests & features which are extracted from the image, an efficient model is derieved.

Step-6: Processed Dataset creation: The data which are processed in previous stages are processed and extracted and converted to a csv file format and stored. This stored data is further utilized for analysis purpose.

Step-7: Training Data Extraction: Randomly the data in csv file is split. The 70% of the split data is used for training the proposed model.

Step-8: Testing Data Extraction: Randomly the data in csv file is split. The 30% of the split data is used for training the proposed model.

Step-9: Classification: Test data has labels such as: Late Blight, Early Blight, and Healthy, based on which classification is performed.

Step-10: Evaluation of proposed model: Depending on the obtained results from the classifier model, the evaluation metrics such as precision, recall, F1-score, and accuracy will be obtained.

(1) (2) (3)

The three Sample leaves of potato are (1): leaf affected by Light Blight (2): leaf affected by Early Blight (3): leaf unaffected (Healthy)

CHAPTER 5

ALGORITHMS

Data preprocessing

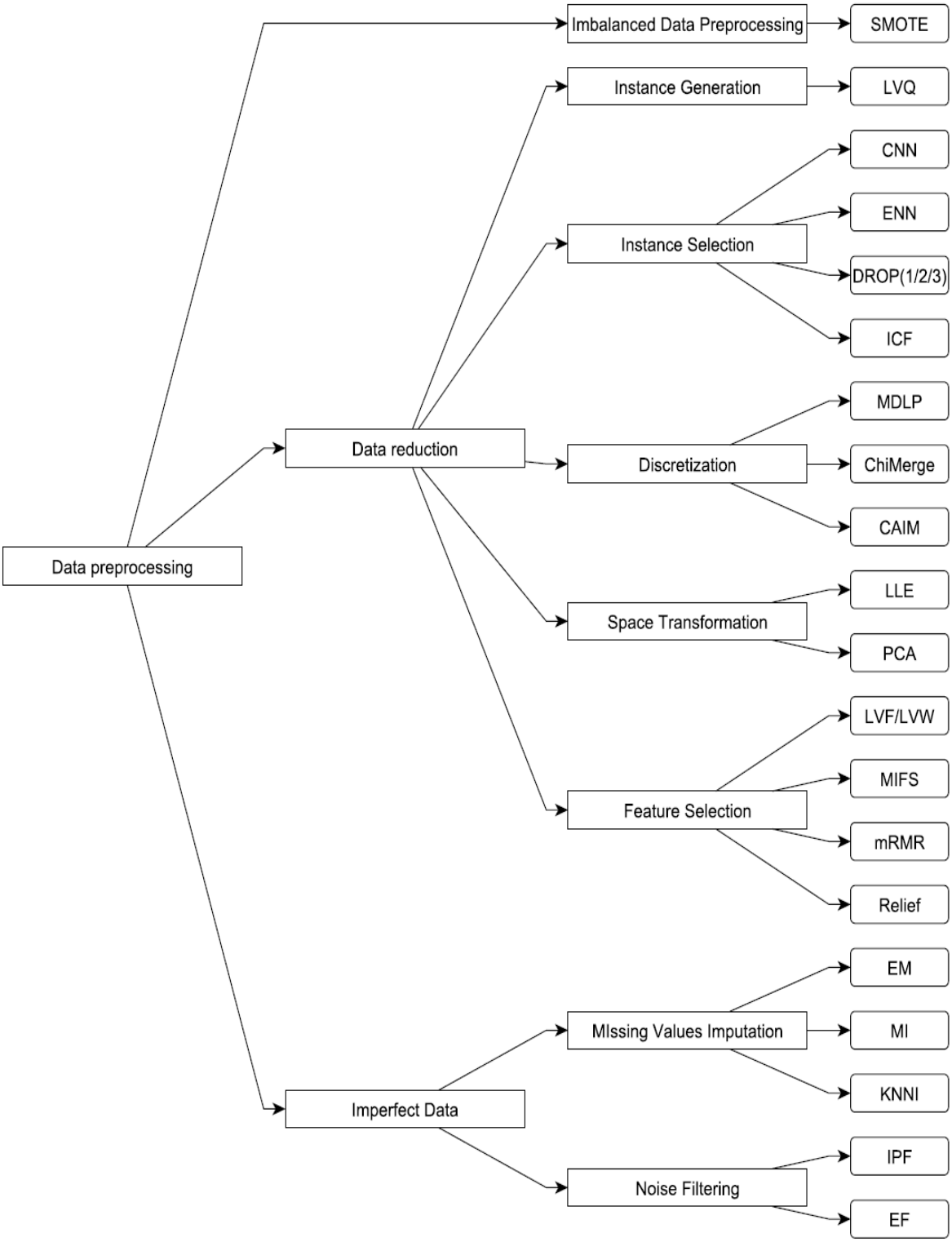
Data preprocessing for Data Mining focuses on one of the most meaningful issues within the famous Knowledge Discovery from Data process. Data will likely have inconsistencies, errors, out of range values, impossible data combinations, missing values or most substantially, data is not suitable to start a DM process. In addition, the growing amount of data in current business applications, science, industry and academia, demands to the requirement of more complex mechanisms to analyze it. With data preprocessing, converting the impractical into possible is achievable, adapting the data to accomplish the input requirements of each DM

algorithm.

The data preprocessing stage can take a considerable amount of processing time. Data preprocessing includes data preparation, compounded by integration, cleaning, normalization and transformation of data; and data reduction tasks, which aim at reducing the complexity of the data, detecting or removing irrelevant and noisy elements from the data through feature selection, instance selection or discretization processes. The outcome expected after a reliable connection of data preprocessing processes is a final data set, which can be contemplated correct and useful for further DM algorithms.

In an effort to identify some of the most influential data preprocessing algorithms that have been widely used in the DM community, we enumerate them according to their usage, popularity and extensions proposed in the research community. We are aware that each

nominated algorithm should have been widely cited and used by other researchers and practitioners in the eld. The selection of the algorithms is based entirely on our criteria and expertise, and it is summarized in the following figure according to their category.


Data preprocessing
Imbalanced Data Preprocessing
SMOTE
Data reduction
Instance Generation
LVQ
Instance Selection
CNN
ENN
DROP(1/2/3)
ICF
Discretization
MDLP
ChiMerge
CAIM
Space Transformation
LLE
PCA
Feature Selection
LVF/LVW
MIFS
mRMR
Relief
Imperfect Data
MIssing Values Imputation
EM
MI
KNNI
Noise Filtering
IPF
EF

The references of such algorithms follow.

Acronym	Reference
EF	C.E. Brodley, M.A. Friedl, Identifying Mislabeled Training Data, Journal of Artificial Intelligence Research 11 (1999) 131{167.
IPF	T.M. Khoshgoftaar, P. Rebours, Improving software quality prediction by noise ltering techniques, Journal of Computer Science and Technology 22 (2007) 387-396.
KNNI	G. Batista, M. Monard, An analysis of four missing data treatment methods for supervised learning, Applied Artificial Intelligence 17 (2003) 519-533.
EM	A. Dempster, N. Laird, D. Rubin, Maximum likelihood estimation from incomplete data via the EM algorithm (with discussion), Journal of the Royal Statistical Society, Series B 39 (1977) 1-38. R.J.A. Little, D.B. Rubin, Statistical Analysis with Missing Data, Wiley Series in Probability and Statistics, Wiley, New York, 1st edition, 1987.
MI	D.B. Rubin, Multiple Imputation for Nonresponse in Surveys, Wiley, 1987.
Relief	K. Kira, L.A. Rendell, A practical approach to feature selection, in: Proceedings of the Ninth International Workshop on Machine Learning, ML92, 1992, pp. 249-256.
mRMR	H. Peng, F. Long, C. Ding, Feature selection based on mutual information: Criteria of maxdependency, max-relevance, and min-redundancy, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (2005) 1226-1238.

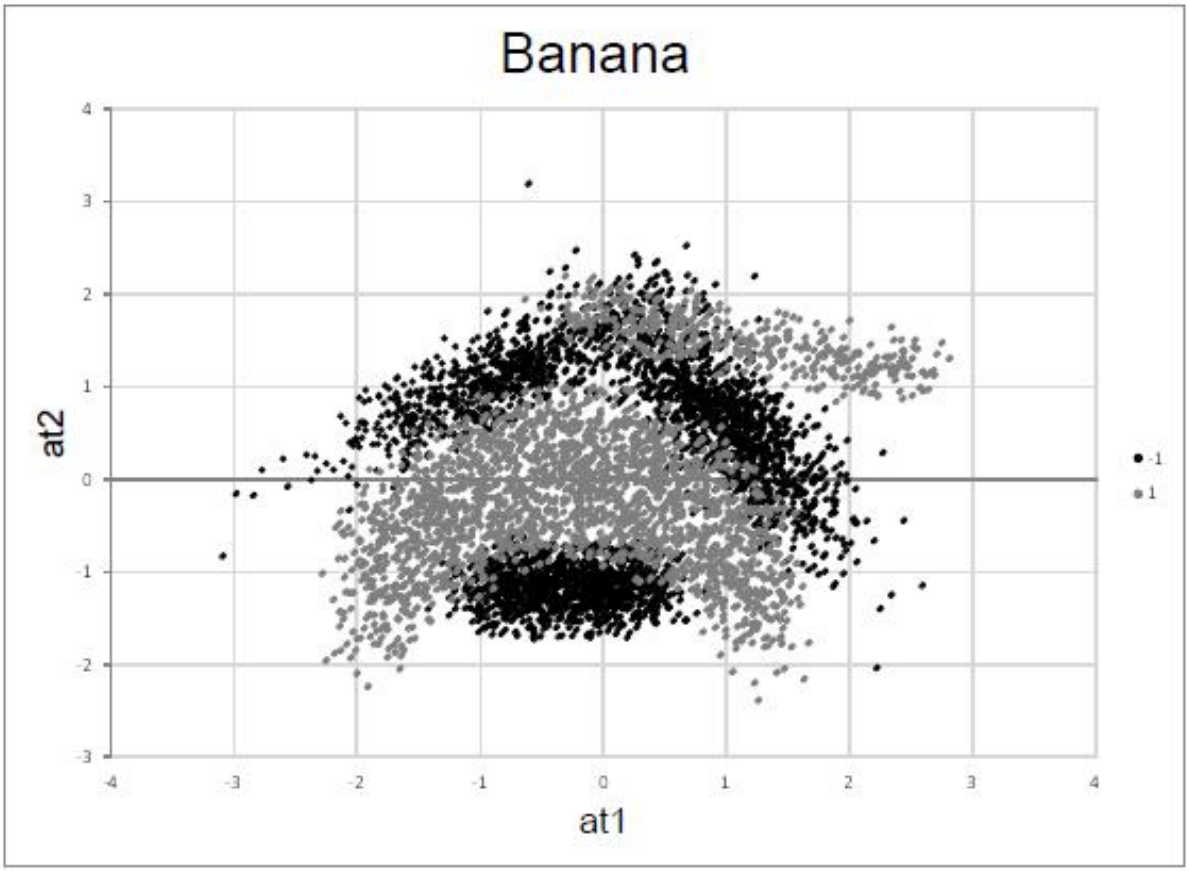
Experimental framework

Two data sets have been used to graphically illustrate the effect of preprocessing techniques:

Banana data set: is a synthetic data set with 5,300 instances, 2 classes and 2 input attributes, which makes it specially well suited to be represented in a plane. As can be seen in the following figure, the classes are not linearly separable, conforming to a "banana" shaped cluster with overlapping between the classes.

Sonar data set: is a real-world data set with 208 instances, 60 real-valued input attributes and 2 classes. Since FS and dimensionally reduction techniques need a large number of input attributes to be useful, we will only use the sonar data set for them.

ECDBL14-20000-N-MV data **set**: is a a reduced sample from the ECDBL'14 Data mining competition (*Data mining competition 2014: Self-deployment track (2014): Evolutionary computation for big data and big learning workshop, 2014*, http://cruncher.ncl.ac.uk/bdcomp/) to a sample with a training fold with 20,000 instances containing MVs and noise. It is coupled with a test set with 20,000 instances. This reduced data set has 50 input attributes (real, integer and nominal valued), with 75% instances of the negative class and 25% instances of the positive class.



Analysis of results:

Sonar and Banana Data Sets

Our goal is also to graphically illustrate the e ect of such techniques, so the reader can visually determine the differences among them by visualizing how they work through graphical representations.

In order to do so, the banana and sonar data sets will be used. Noise, imbalance or missing values are artificially introduced in the sonar data set to enable the usage of imputation and filters. For those techniques aimed to reduce the dimensionality, the sonar data set is used instead.

Preprocessing type
Missing values (Banana data set) We have introduced a 10% of missing values (MVs) in the data set. Then imputation methods (KNNI, EM and MI) are used to estimate them.
Noise (Banana data set) We have introduced a 10% of noise in the data set. Then filtering methods (EF and IPF) are used to filter the noisy instances.

Feature selection (Sonar data set)

Several feature selection methods are used to reduce the dimensionality of the sonar data set.

To facilitate the visualization of the resulting data sets, PCA is applied in order to obtain a 2D projection of them.

Space transformations (Sonar data set)

We apply PCA and LLE to transform the sonar data set to a 2D transformed version.

Instance reduction (Banana data set)

We use several instance selection algorithms to reduce the number of instances of the banana data set.

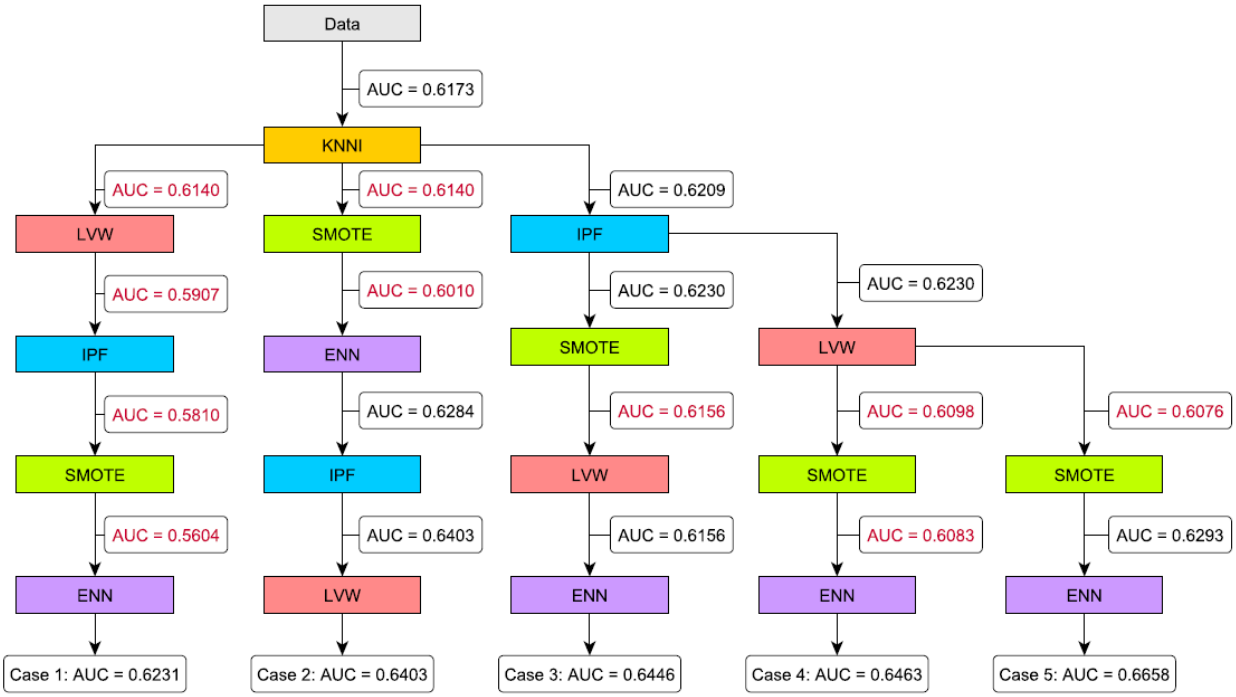
Imbalanced learning (Banana data set)

First, we apply Random Undersampling (RUS) to create an imbalanced version of the banana data set.

Then SMOTE is used to balance the class distribution, generating synthetic examples of the minority class.

In order to carry out the analysis, we will use the ECDBL14-20000-N-MV data set with missing values, noise and an imbalanced class distribution. We have selected one preprocessing algorithm of each type to be applied to this data set:

- an imputation method (kNNI),
- a noise ltering algorithm (IPF),
- a FS method (LVW),an instance selector (ENN) and
- the over-sampling algorithm for imbalanced data sets (SMOTE).



As we can observe from the different cases, the order in which the preprocessing techniques are applied has a large impact in the final result. It is also important to notice that some preprocessing techniques may seem counterproductive at first glance, but they will finally yield a benefit when combined with others.

Classification

Once the features have been extracted, then these features are to be used to classifyand identify an object using SVM classifier to classify plants based on shape-related features of leaf such as aspect ratio, rectangularity, area ratio of convex hull, perimeter ratio of convex hull, sphericity, circularity, eccentricity, form factor and invariant moments.

In general pattern recognition systems, there are two steps in building a classifier: training and testing (or recognition). These steps can be further broken down into sub-steps.

Training:

1. Pre-processing: Process the data so it is in a suitable form.
2. Feature extraction: Reduce the amount of data by extracting relevant information, usually results in a vector of scalar values.
3. Model Estimation: From the finite set of feature vectors, need to estimate a model (usually statistical) for each class of the training data.

Recognition:

1. Pre-processing:
2. Feature extraction: (both steps are same as above)
3. Classification: Compare feature vectors to the various models and find the closest match. One can match the feature vectors obtained in training set.

The algorithm has three main parts: Training, Classification, Segmentation and distance measurement.

SUPPORT VECTOR MACHINE (SVM) CLASSIFIER

SVMs (Support Vector Machines) are a useful technique for data classification. Classification task usually involves separating data into training and testing sets. Each instance in the training set contains one \target value" (i.e. the class labels) and several attributes" (i.e. the

features or observed variables). The goal of SVM is to produce a model (based on the training data) which predicts the target values of the test data given only the test data attributes.A Support Vector Machine (SVM) is a discriminative classifier formally defined by separating hyperplane. In other words, given labeled training data (*supervised learning*), the algorithm outputs an optimal hyperplane which categorizes new examples.

Let's consider the following simple problem: We are given a set of n points (vectors) : x1, x2, x3, ….. xn such that xi is a vector of length m, and each belong to one of two classes we label them by "+1" and "-1". So our training set is

$$(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$$
$$\forall i \quad x_i \in R^m, \; y_i \in \{+1, -1\}$$

We want to find a separating hyperplane that separates these points into the two classes. "The positives" (class "+1") and "The negatives" (class "-1"). Let's introduce the notation used to define formally a hyperplane:

$$f(x) = \beta_0 + \beta^T x,$$

Where, β is known as the *weight vector* and β_0 as the *bias*. For a linearly separable set of 2D points which belong to one of two classes, find a separating straight line.

Figure 7.1, there exist multiple lines that offer a solution to the problem.

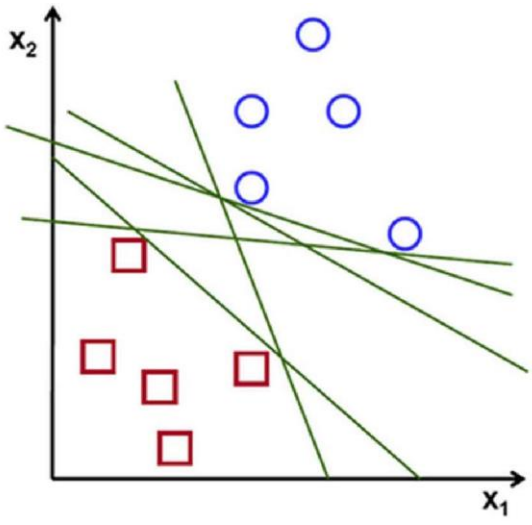


Fig. 5.3 Green color hyperplane separating two classes of red squares and blue circles.

A line is bad if it passes too close to the points because it will be noise sensitive and it will not generalize correctly. Therefore, our goal should be to find the line passing as far as possible from all points.

Then, the operation of the SVM algorithm is based on finding the hyperplane tha gives the largest minimum distance to the training examples. Twice, this distance receives the important name of **margin** within SVM's theory. Therefore, the optimal separating hyperplane *maximizes* the margin of the training data which is depicted well in the Figure 7.2.

The optimal hyperplane can be represented in an infinite number of different ways by scaling of β and β_0. As a matter of convention, among all the possible representations of the hyperplane, the one chosen is

$$|\beta_0 + \beta^T x| = 1$$

Where symbolizes the training examples closest to the hyperplane. In general, the training examples that are closest to the hyperplane are called **support vectors**. This representation is known as the **canonical hyperplane**.

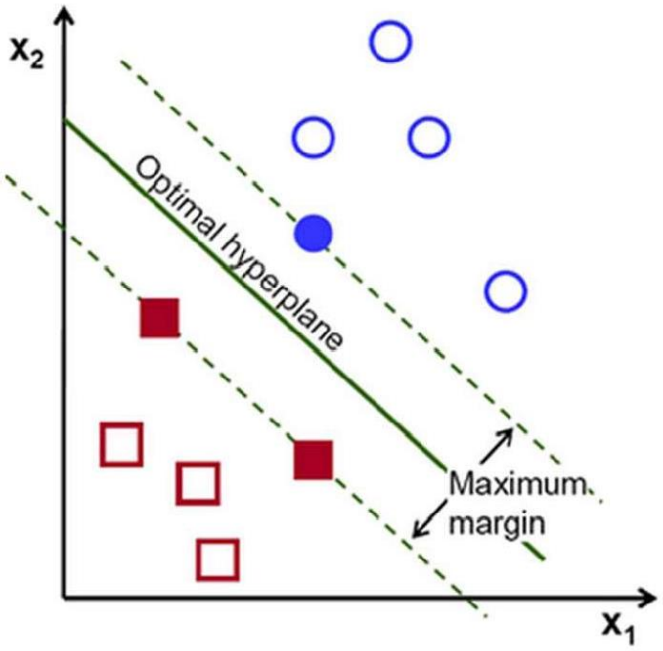


Figure 5.4 Finding an optimal hyperplane.

In particular, by comparing with the 1-NN and k-NN classifiers [11-16], it can be found that the SVM classifier can not only save the storage space but also reduce the classification time under the case of no sacrificing the classification accuracy.

Evaluating a Classification model:

Once our model is completed, it is necessary to evaluate its performance; either it is a Classification or Regression model. So for evaluating a Classification model, we have the following ways:

1. Log Loss or Cross-Entropy Loss:

- It is used for evaluating the performance of a classifier, whose output is a probability value between the 0 and 1.
- For a good binary Classification model, the value of log loss should be near to 0.
- The value of log loss increases if the predicted value deviates from the actual value.
- The lower log loss represents the higher accuracy of the model.
- For Binary classification, cross-entropy can be calculated as:

1. ?(ylog(p)+(1?y)log(1?p))

Where y= Actual output, p= predicted output.

2. Confusion Matrix:

- The confusion matrix provides us a matrix/table as output and describes the performance of the model.
- It is also known as the error matrix.
- The matrix consists of predictions result in a summarized form, which has a total number of correct predictions and incorrect predictions. The matrix looks like as below table:

	Actual Positive	**Actual Negative**

Predicted Positive	True Positive	False Positive
Predicted Negative	False Negative	True Negative

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{Total Population}}$$

MODULES

PIL – Python Image Library

The **Python Imaging Library** adds image processing capabilities to your Python interpreter. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.

The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.

Let's look at a few possible uses of this library.

Image Archives

The Python Imaging Library is ideal for image archival and batch processing applications. You can use the library to create thumbnails, convert between file formats, print images, etc.

The current version identifies and reads a large number of formats. Write support is intentionally restricted to the most commonly used interchange and presentation formats.

Image Display

The current release includes Tk PhotoImage and BitmapImage interfaces, as well as a Windows DIB interface that can be used with PythonWin and other Windows-based toolkits. Many other GUI toolkits come with some kind of PIL support.

For debugging, there's also a show() method which saves an image to disk, and calls an external display utility.

Image Processing

The library contains basic image processing functionality, including point operations, filtering

with a set of built-in convolution kernels, and colour space conversions.
The library also supports image resizing, rotation and arbitrary affine transforms.
There's a histogram method allowing you to pull some statistics out of an image. This can be used for automatic contrast enhancement, and for global statistical analysis.

PyTorch

PyTorch provides the elegantly designed modules and classes torch.nn , torch.optim , Dataset , and DataLoader to help you create and train neural networks. In order to fully utilize their power and customize them for your problem, you need to really understand exactly what they're doing. To develop this understanding, we will first train basic neural net on the MNIST data set without using any features from these models; we will initially only use the most basic PyTorch tensor functionality. Then, we will incrementally add one feature from `torch.nn`, `torch.optim`, `Dataset`, or `DataLoader` at a time, showing exactly what each piece does, and how it works to make the code either more concise, or more flexible.

Torchvision

What is torchvision?

Torchvision is a library for Computer Vision that goes hand in hand with PyTorch. It has utilities for efficient Image and Video transformations, some commonly used pre-trained models, and some datasets (torchvision does not come bundled with PyTorch, you will have to install it separately.)

This library is part of the PyTorch project. PyTorch is an open source machine learning framework.
Features described in this documentation are classified by release status:
Stable: These features will be maintained long-term and there should generally be no major performance limitations or gaps in documentation. We also expect to maintain backwards compatibility (although breaking changes can happen and notice will be given one release ahead of time).
Beta: Features are tagged as Beta because the API may change based on user feedback, because the performance needs to improve, or because coverage across operators is not yet complete. For Beta features, we are committing to seeing the feature through to the Stable classification.
Prototype: These features are typically not available as part of binary distributions like PyPI or Conda, except sometimes behind run-time flags, and are at an early stage for feedback and testing.
The torchvision package consists of popular datasets, model architectures, and common image

transformations for computer vision.

TRANSFORMING AND AUGMENTING IMAGES

Transforms are common image transformations available in the torchvision.transforms module. They can be chained together using Compose. Most transform classes have a function equivalent: functional transforms give fine-grained control over the transformations. This is useful if you have to build a more complex transformation pipeline (e.g. in the case of segmentation tasks).

Most transformations accept both PIL images and tensor images, although some transformations are PIL-only and some are tensor-only. The Conversion Transforms may be used to convert to and from PIL images.

The transformations that accept tensor images also accept batches of tensor images. A Tensor Image is a tensor with (C, H, W) shape, where C is a number of channels, H and W are image height and width. A batch of Tensor Images is a tensor of (B, C, H, W) shape, where B is a number of images in the batch. The expected range of the values of a tensor image is implicitly defined by the tensor dtype. Tensor images with a float dtype are expected to have values in [0, 1). Tensor images with an integer dtype are expected to have values in [0, MAX_DTYPE] where MAX_DTYPE is the largest value that can be represented in that dtype.

Randomized transformations will apply the same transformation to all the images of a given batch, but they will produce different transformations across calls. For reproducible transformations across calls, you may use functional transforms.

Models And Pre-Trained Weights

The torchvision.models subpackage contains definitions of models for addressing different tasks, including: image classification, pixelwise semantic segmentation, object detection, instance segmentation, person keypoint detection, video classification, and optical flow.

General information on pre-trained weights

TorchVision offers pre-trained weights for every provided architecture, using the PyTorch torch.hub. Instancing a pre-trained model will download its weights to a cache directory. This directory can be set using the *TORCH_HOME* environment variable. See torch.hub.load_state_dict_from_url() for details.

Using the pre-trained models

Before using the pre-trained models, one must preprocess the image (resize with right resolution/interpolation, apply inference transforms, rescale the values etc). There is no standard way to

do this as it depends on how a given model was trained. It can vary across model families, variants or even weight versions. Using the correct preprocessing method is critical and failing to do so may lead to decreased accuracy or incorrect outputs.

All the necessary information for the inference transforms of each pre-trained model is provided on its weights documentation. To simplify inference, TorchVision bundles the necessary preprocessing transforms into each model weight.

What is so good with torchvision ?

- Since it is an accompaniment to PyTorch, it automatically comes with the **GPU support**. (**So, it is FAST !**)
- Its development philosophy is to be simple in implementation (eg: without an extensive argument set for its functions) . The developers have kept it separately from PyTorch to keep it lean and lightweight.
- It is **ready to use** ! (comes with sample data-set (CIFAR10, CelebA etc) , some commonly used pre-trained models (ResNet18, maskRCNN_resnet50 etc) and even sample starter codes for some of the typical AI/ Machine Learning Problems (Image Classification, Semantic Segmentation , Keypoint detection etc) — all inbuilt into its library
- It is developed and maintained by the Facebook AI team, and supported by the python community.

1. Datasets

torchvision comes with an option to readily download a set of most commonly used datasets — more than enough to get you started to most of your implementations . It downloads the datasets onto your local system (first time only) and then from here on — you can directly use that in your program by referencing it .

Some of the supported datasets are —

- CIFAR
- CelebA
- COCO
- Omniglot
- VOC

- Flickr
- FashionMNIST

2. Pre-trained Models

To facilitate transfer learning, torch vision has specific pre-trained models for Image classification , Object Detection , Instance segmentation and even Video Classification. It downloads the models on to you local system (first time only).
some of these pre-trained models available are —

- ResNet 3D 18 (video classification)
- MASK R-CNN (Instance Segmentation)
- AlexNet , VGG, ResNet, Inception etc (for typical Image classification problems)

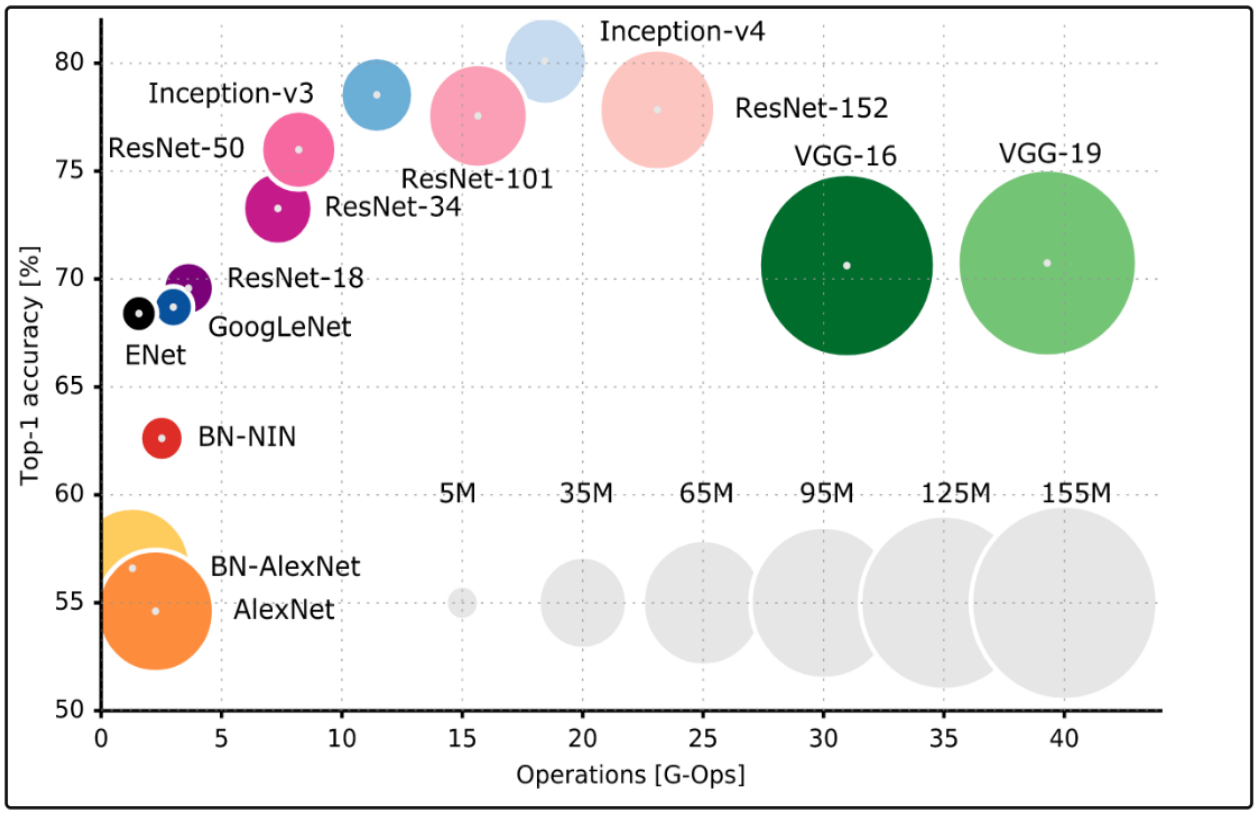
Here is a comparison on different models for G-ops v/s accuracy

Torch.nn.functional

The first and easiest step is to make our code shorter by replacing our hand-written activation and loss functions with those from `torch.nn.functional` (which is generally imported into the namespace `F` by convention). This module contains all the functions in the `torch.nn` library (whereas other parts of the library contain classes). As well as a wide range of loss and activation functions, you'll also find here some convenient functions for creating neural nets, such as pooling functions. (There are also functions for doing convolutions, linear layers, etc, but as we'll see, these are usually better handled using other parts of the library.)

If you're using negative log likelihood loss and log softmax activation, then Pytorch provides a single function `F.cross_entropy` that combines the two. So we can even remove the activation function from our model.

`nn.Module` and `nn.Parameter`, for a clearer and more concise training loop. We subclass `nn.Module` (which itself is a class and able to keep track of state). In this case, we want to create a class that holds our weights, bias, and method for the forward step. `nn.Module` has a number of attributes and methods (such as `.parameters()` and `.zero_grad()`) which we will be using.



Different pre-trained models (Source : https://culurciello.github.io/tech/2016/06/20/training-enet.html)

3. Transforms

Transforms are easily callable modules within the library that are used for Image Augmentations and transformations

Common/Important transforms are

-> ToTensor() — Convert an Image datasets to Tensors

-> CenterCrop() — Crops with the center fixed

-> Normalize() — Normalize the pixel values to that of the dataset that you are using

-> Pad() — to give a padding or border

Data loading order and sampler

For iterable-style datasets, data loading order is entirely controlled by the user-defined iterable. This allows easier implementations of chunk-reading and dynamic batch size (e.g., by yielding a batched sample at each time).

The rest of this section concerns the case with map-style datasets. torch.utils.data.Sampler classes are used to specify the sequence of indices/keys used in data loading. They represent iterable objects over the indices to datasets. E.g., in the common case with stochastic gradient decent (SGD), a Sampler could randomly permute a list of indices and yield each one at a time, or yield a small number of them for mini-batch SGD.

A sequential or shuffled sampler will be automatically constructed based on the shuffle argument to a DataLoader. Alternatively, users may use the sampler argument to specify a custom Sampler object that at each time yields the next index/key to fetch.

A custom Sampler that yields a list of batch indices at a time can be passed as the batch_sampler argument. Automatic batching can also be enabled via batch_size and drop_last arguments.

Transforming and Augmenting Images

Transforms are common image transformations available in the torchvision.transforms module. They can be chained together using Compose. Most transform classes have a function equivalent: functional transforms give fine-grained control over the transformations. This is useful if you have to build a more complex transformation pipeline (e.g. in the case of segmentation tasks).

Most transformations accept both PIL images and tensor images, although some transformations are PIL-only and some are tensor-only. The Conversion Transforms may be used to convert to and from PIL images.

The transformations that accept tensor images also accept batches of tensor images. A Tensor Image is a tensor with (C, H, W) shape, where C is a number of channels, H and W are image height and width. A batch of Tensor Images is a tensor of (B, C, H, W) shape, where B is a number of images in the batch.

The expected range of the values of a tensor image is implicitly defined by the tensor dtype. Tensor images with a float dtype are expected to have values in [0, 1). Tensor images with an integer dtype are expected to have values in [0, MAX_DTYPE] where MAX_DTYPE is the largest value that can be represented in that dtype.

Randomized transformations will apply the same transformation to all the images of a given batch, but they will produce different transformations across calls. For reproducible transformations across calls, you may use functional transforms.

Conversion transforms

ToPILImage([mode])	Convert a tensor or an ndarray to PIL Image.
ToTensor()	Convert a PIL Image or numpy.ndarray to tensor.
PILToTensor()	Convert a PIL Image to a tensor of the same type.

CHAPTER 6

DATASET

The dataset acquired is for Identification of Plant Leaf Diseases Using a 9-layer Deep Convolutional Neural Network

In this data-set, 39 different classes of plant leaf and background images are available. The data-set containing 61,486 images. We used six different augmentation techniques for increasing the data-set size. The techniques are image flipping, Gamma correction, noise injection, PCA color augmentation, rotation, and Scaling.

The classes are,

1.Apple_scab

2.Apple_black_rot

3.Apple_cedar_apple_rust

4.Apple_healthy

5.Background_without_leaves

6.Blueberry_healthy

7.Cherry_powdery_mildew

8.Cherry_healthy

9.Corn_gray_leaf_spot

10.Corn_common_rust

11.Corn_northern_leaf_blight

12.Corn_healthy

13.Grape_black_rot

14.Grape_black_measles

15.Grape_leaf_blight

16.Grape_healthy

17.Orange_haunglongbing

18.Peach_bacterial_spot

19.Peach_healthy

20.Pepper_bacterial_spot

21.Pepper_healthy

22.Potato_early_blight

23.Potato_healthy

24.Potato_late_blight

25.Raspberry_healthy

26.Soybean_healthy

27.Squash_powdery_mildew

28.Strawberry_healthy

29.Strawberry_leaf_scorch

30.Tomato_bacterial_spot

31.Tomato_early_blight

32.Tomato_healthy

33.Tomato_late_blight

34.Tomato_leaf_mold

35.Tomato_septoria_leaf_spot

36.Tomato_spider_mites_two-spotted_spider_mite

37.Tomato_target_spot

38.Tomato_mosaic_virus

39.Tomato_yellow_leaf_curl_virus

The current dataset was further fissured into two datasets to check the accuracy of the model by using augmented and non-augmented dataset. The following are the two datasets

- Plant_leaf_diseases_dataset_without_augmentation; To check the accuracy without augmentation.
- Plant_leaf_diseases_dataset_with_augmentation: To check the accuracy with augmentation.

RESULTS

Disease detection using the Model:

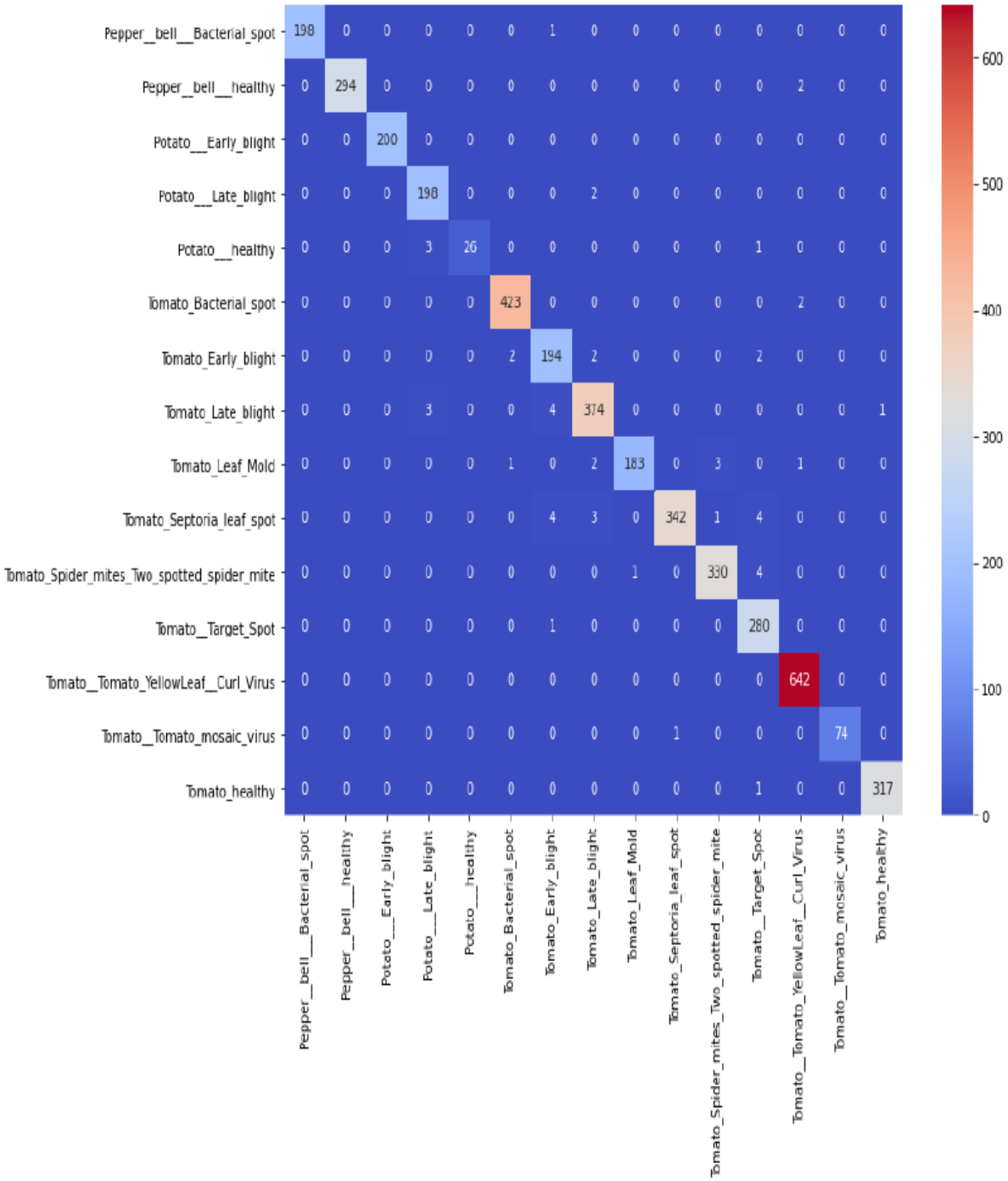
At first machine learning algorithm was implemented to detect plant disease. This was done in two phases. I.) Implementing ML algorithm on potato dataset. II) Implementing ML algorithm on the entire dataset. (currently not done in this project).

I. Implementing ML algorithm on potato dataset.

The dataset which is considered in the proposed work is an openly accessed dataset & it was randomly divided into the training dataset consists of 1820 images and the testing dataset consists of 780 images. The otsu algorithm was utilized for the binary image segmentation and infected region identification this was done with the help of preparing an image mask. The Gray Level Co-occurrence Matrix is the main tool that implements the concepts learned from extracted features. utilized for feature extraction, & multi-class support vector machine(SVM) methodology was utilized for the classification of potato leaves. The model derived is evaluated using certain evaluation metrics: precision, recall, F1-score, and accuracy.

Evaluation Metrics of Potato Disease analysis using SVM

Category Name	Precision (%)	Recall(%)	Fl-score (%)	Accuracy (%)
Late Blight	91.07	95.41	93.29	94.71
Early Blight	98.36	94.71	96.43	96.84
Healthy	98.93	98.62	98.76	96.43
Overall	96.12	96.25	96.16	95.99



Anomalous spikes is seen if the crop had been contradicted with an infection/disease

ADVANTAGES AND APPLICATIONS

Advantages:

Disease detection using Image Processing techniques helps to detect the disease in very initial stage. In previous days, farmers need to take the samples and go to labs for detecting the plant disease. This requires money and it is time consuming.

Image processing method helps to detect the image quickly. It even requires less money. By knowing the plant disease in early stages, it can be cured by taking appropriate measures. This method helps farmers during their daily struggle against plant disease outbreaks.

Applications:

Few Image Processing Applications are:

- **Image Sharpening and Restoration:**

Image sharpening and restoration refers here to process images that have been captured from the modern camera to make them a better image or to manipulate those images in way to achieve desired result. It refers to do what Photoshop usually does. This include zooming, blurring, sharpening, gray scale to colour conversion, detecting edges, Image retrieval and Image recognition.

- **Medical Field:**

The common applications of DIP in the field of medical is

  - Gamma ray imaging
  - PET scan
  - X Ray imaging
  - Medical CT
  - UV imaging

- **Colour Processing:**

Colour processing includes processing of coloured images and different color spaces that are used. For example RGB colour model, YCbCr, HSV. It also involves studying transmission, storage, and encoding of these colour images.

- **Pattern Recognition:**

Here recognition involves study from image processing and from various other fields that include machine learning(a branch of artificial intelligent).In pattern recognitions,

image processing is used for identifying the objects in the images and then machine learning is used to train the system for the change in pattern. Pattern recognition is used in computer aided diagnosis, recognition of handwriting, recognition of images etc.

- **Video Processing :**

  A video is nothing but just the very fast movement of pictures. The quality of the video depends on the number of frames/pictures per minute and the quality of each frame being used. Video processing involves noise reductions, detail enchancements, motion detection, frame rate conversion, aspect ratio conversion, colour space conversion etc.

- **Machine/Robot Vision**

  Apart from many challenges that a robot face today, the biggest challenge is to increase the vision of the robot. Make robot able to see things, identify them, identify the hurdles etc image processing techniques are applied. Much work has been contributed by this field and a complete other field of computer vision has been introduced to work on it.

- **Hurdle Detection**

  Hurdle detection is one of the common tasks that has been done through image processing by identifying different type of objects in the image and then calculating the distance between robot and hurdles.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

An automatic detection application for plant disease recognition was presented. It is based on image processing that analyzes the color features of the spots in plant parts. It was evaluated on crop diseases with an accuracy that exceeds 90% using a small training set. The application has been tested here for farmland diseases based on photographs of leaves. A success rate higher than 90% has been achieved in the disease recognition process. The quantified features used for the recognition of the supported plant diseases can be announcedto the user making the system easily extendible to new diseases.

Image segmentation is mainly performed on the leaves in order to check the disease present in it. The pathogens are basically responsible for the plant disease which destroys the leaf and stem of the plants. Back propagation (BP) networks, radial basis function (RBF) neural networks, generalized regression networks (GRNNs) and probabilistic neural networks (PNNs), image enhancement, image segmentation, feature extraction, fuzzy logic, Multi-class Support Vector Machine and Local Binary Pattern are used as the techniques which helps in the detection ofthe plant diseases. Many plant diseases and different approaches for their detection are given in detail. In future many other techniques can be used.

In the future, the proposed methodology can be integrated with other yet to be developed, methods for disease identification and classification using color and texture analysis to develop an expert system for detecting the plant disease for warning and administration, where the disease type can be identified by color and texture analysis and the severity level estimation by our proposed method since itis disease independent. The performance of the system can be improved in the future by using advanced background separation methods to separate the leaf object from a complex background. More infections like downy mildew (DM) and sudden death syndrome (SDS) can also be classified along with the BB, FE, BS, and SR by using the cataloguing algorithm, but due to non-availability of suitable and sufficient training and test data at present it has not been incorporated into the present work. The similar methodology can be appliedto other plant foliar infections and early warning systems for - 165 - rice, cotton- crops, fruits, vegetables and beans, etc. The use of other cataloguing methods can be exploited to improve the accuracy of the system in future.

REFERENCES

[1] M.B. Riley, M.R. Williamson, and O. Maloy, “Plant disease diagnosis,” The Plant Health Instructor,2002.doi=10.1094/PHI-I-2002-1021-01.

[2] S. Sankaran, A. Mishra, R. Eshani and C. Davis, “A review of advanced techniques for detecting plant diseases,” Computers and Electronics in Agriculture, vol. 72, no. 1, pp. 1-13, June,2010.

[3] N.W. Schaad and R.D. Frederick, “Real time PCR and its application for rapid plant disease diagnostics,” Canadian Journal of Plant Pathology, vol. 24, no. 3, pp. 250-258, July 2002.

[4] K. Georgakopoulou, C. Spathis, N. Petrellis and A. Birbas “A Capacitive to digital Converter with Automatic Range Adaptation,” IEEE Trans. On Instrumentation and Measurements,vol.65,no.2,pp.336-345,Feb2016.

[5] D.E. Purcell, M.G. O’ Shea, R.A. Johnson and S. Kokot, “Near infrared spectroscopy for the prediction of disease rating for Fiji leaf gall in sugarcane clones”, Applied Spectroscopy, vol. 63,no.4,pp.450-457,Apr2009.

[6] G.C.A. Barbedo, “Digital image processing techniques for detecting quantifying and classifyingplantdiseases,”SpringerPlus,2:660,Dec.2013.

[7] A. Kulkarni and A. Patil, “Applying image processing technique to detect plant diseases,” International Journal of Modern Engineering Research, vol. 2, no. 5, pp. 3361-3364, Oct 2012.

[8] C. Mix, F.X. Picó and N.J. Ouborg, “A Comparison of Stereomicroscope And Image Analysis For Quantifying Fruit Traits,” SEED Technology vol.25,no.1.

[9] S.S. Abu-Naser, K.A. Kashkash and M. Fayyad, “Developing an Expert System for Plant Disease Diagnosis,” Asian Network for Scientific Information, Journal of Artificial Intelligence,

vol.1,no.2,pp.78-85,2008.

[10] J.C. Lai, B. Ming, S.K. Li, K.R. Wang, R.Z. Xie and S.J. Gao, “An Image-Based Diagnostic Expert System for Corn Diseases,” Elsevier Agricultural Sciences in China, vol. 9, no. 8, pp. 1221-1229, Aug. 2010

[11] Plant Disease Detection using Digital Image Processing and GSM by Pallavi.S.Marathe 2017 IJESC, Volume 7 Issue No.4

APPENDIX

CROP.ipynb

```
# %% [markdown]
# ### Import Dependencies

# %%
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# %%
import torch
from torchvision import datasets, transforms, models  # datsets  , transforms
from torch.utils.data.sampler import SubsetRandomSampler
import torch.nn as nn
import torch.nn.functional as F
from datetime import datetime

# %%
# %load_ext nb_black

# %% [markdown]
# ### Import Dataset

# %% [markdown]
# <b> Dataset Link (Plant Vliiage Dataset ):</b><br> <a
href='https://data.mendeley.com/datasets/tywbtsjrjv/1'>
https://data.mendeley.com/datasets/tywbtsjrjv/1 </a>

# %%
transform = transforms.Compose(
    [transforms.Resize(255), transforms.CenterCrop(224), transforms.ToTensor()]
```

```
)

# %%
dataset = datasets.ImageFolder("Dataset", transform=transform)

# %%
dataset

# %%
indices = list(range(len(dataset)))

# %%
split = int(np.floor(0.85 * len(dataset)))  # train_size

# %%
validation = int(np.floor(0.70 * split))  # validation

# %%
print(0, validation, split, len(dataset))

# %%
print(f"length of train size :{validation}")
print(f"length of validation size :{split - validation}")
print(f"length of test size :{len(dataset)-validation}")

# %%
np.random.shuffle(indices)

# %% [markdown]
# ### Split into Train and Test

# %%
train_indices, validation_indices, test_indices = (
```

```
    indices[:validation],
    indices[validation:split],
    indices[split:],
)

# %%
train_sampler = SubsetRandomSampler(train_indices)
validation_sampler = SubsetRandomSampler(validation_indices)
test_sampler = SubsetRandomSampler(test_indices)

# %%
targets_size = len(dataset.class_to_idx)

# %% [markdown]
# ### Model

# %% [markdown]
# <b>Convolution Aithmetic Equation : </b>(W - F + 2P) / S + 1 <br>
# W = Input Size<br>
# F = Filter Size<br>
# P = Padding Size<br>
# S = Stride <br>

# %% [markdown]
# ### Transfer Learning

# %%
# model = models.vgg16(pretrained=True)

# %%
# for params in model.parameters():
#     params.requires_grad = False
```

```
# %%
# model

# %%
# n_features = model.classifier[0].in_features
# n_features

# %%
# model.classifier = nn.Sequential(
#     nn.Linear(n_features, 1024),
#     nn.ReLU(),
#     nn.Dropout(0.4),
#     nn.Linear(1024, targets_size),
# )

# %%
# model

# %% [markdown]
# ### Original Modeling

# %%
class CNN(nn.Module):
    def _init_(self, K):
        super(CNN, self)._init_()
        self.conv_layers = nn.Sequential(
            # conv1
            nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(32),
            nn.Conv2d(in_channels=32, out_channels=32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(32),
```

```
    nn.MaxPool2d(2),
    # conv2
    nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, padding=1),
    nn.ReLU(),
    nn.BatchNorm2d(64),
    nn.Conv2d(in_channels=64, out_channels=64, kernel_size=3, padding=1),
    nn.ReLU(),
    nn.BatchNorm2d(64),
    nn.MaxPool2d(2),
    # conv3
    nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, padding=1),
    nn.ReLU(),
    nn.BatchNorm2d(128),
    nn.Conv2d(in_channels=128, out_channels=128, kernel_size=3, padding=1),
    nn.ReLU(),
    nn.BatchNorm2d(128),
    nn.MaxPool2d(2),
    # conv4
    nn.Conv2d(in_channels=128, out_channels=256, kernel_size=3, padding=1),
    nn.ReLU(),
    nn.BatchNorm2d(256),
    nn.Conv2d(in_channels=256, out_channels=256, kernel_size=3, padding=1),
    nn.ReLU(),
    nn.BatchNorm2d(256),
    nn.MaxPool2d(2),
)

self.dense_layers = nn.Sequential(
    nn.Dropout(0.4),
    nn.Linear(50176, 1024),
    nn.ReLU(),
    nn.Dropout(0.4),
    nn.Linear(1024, K),
```

```
        )

    def forward(self, X):
        out = self.conv_layers(X)

        # Flatten
        out = out.view(-1, 50176)

        # Fully connected
        out = self.dense_layers(out)

        return out

# %%
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(device)

# %%
device = "cpu"

# %%
model = CNN(targets_size)

# %%
model.to(device)

# %%
from torchsummary import summary

summary(model, (3, 224, 224))

# %%
criterion = nn.CrossEntropyLoss()  # this include softmax + cross entropy loss
```

```
optimizer = torch.optim.Adam(model.parameters())

# %% [markdown]
# ### Batch Gradient Descent

# %%
def batch_gd(model, criterion, train_loader, test_laoder, epochs):
    train_losses = np.zeros(epochs)
    test_losses = np.zeros(epochs)

    for e in range(epochs):
        t0 = datetime.now()
        train_loss = []
        for inputs, targets in train_loader:
            inputs, targets = inputs.to(device), targets.to(device)

            optimizer.zero_grad()

            output = model(inputs)

            loss = criterion(output, targets)

            train_loss.append(loss.item())  # torch to numpy world

            loss.backward()
            optimizer.step()

        train_loss = np.mean(train_loss)

        validation_loss = []

        for inputs, targets in validation_loader:
```

```
            inputs, targets = inputs.to(device), targets.to(device)

            output = model(inputs)

            loss = criterion(output, targets)

            validation_loss.append(loss.item())  # torch to numpy world

        validation_loss = np.mean(validation_loss)

        train_losses[e] = train_loss
        validation_losses[e] = validation_loss

        dt = datetime.now() - t0

        print(
            f"Epoch : {e+1}/{epochs} Train_loss:{train_loss:.3f} Test_loss:{validation_loss:.3f}
Duration:{dt}"
        )

    return train_losses, validation_losses

# %%
device = "cpu"

# %%
batch_size = 64
train_loader = torch.utils.data.DataLoader(
    dataset, batch_size=batch_size, sampler=train_sampler
)
test_loader = torch.utils.data.DataLoader(
    dataset, batch_size=batch_size, sampler=test_sampler
)
validation_loader = torch.utils.data.DataLoader(
```

```
    dataset, batch_size=batch_size, sampler=validation_sampler
)

# %%
train_losses, validation_losses = batch_gd(
    model, criterion, train_loader, validation_loader, 5
)

# %% [markdown]
# ### Save the Model

# %%
# torch.save(model.state_dict() , 'plant_disease_model_1.pt')

# %% [markdown]
# ### Load Model

# %%
targets_size = 39
model = CNN(targets_size)
model.load_state_dict(torch.load("plant_disease_model_1_latest.pt"))
model.eval()

# %%
# %matplotlib notebook

# %% [markdown]
# ### Plot the loss

# %%
plt.plot(train_losses , label = 'train_loss')
plt.plot(validation_losses , label = 'validation_loss')
plt.xlabel('No of Epochs')
```

```
plt.ylabel('Loss')
plt.legend()
plt.show()

# %% [markdown]
# ### Accuracy

# %%
def accuracy(loader):
    n_correct = 0
    n_total = 0

    for inputs, targets in loader:
        inputs, targets = inputs.to(device), targets.to(device)

        outputs = model(inputs)

        _, predictions = torch.max(outputs, 1)

        n_correct += (predictions == targets).sum().item()
        n_total += targets.shape[0]

    acc = n_correct / n_total
    return acc

# %%
train_acc = accuracy(train_loader)
test_acc = accuracy(test_loader)
validation_acc = accuracy(validation_loader)

# %%
print(
    f"Train Accuracy : {train_acc}\nTest Accuracy : {test_acc}\nValidation Accuracy :
{validation_acc}"
```

```
)

# %% [markdown]
# ### Single Image Prediction

# %%
transform_index_to_disease = dataset.class_to_idx

# %%
transform_index_to_disease = dict(
    [(value, key) for key, value in transform_index_to_disease.items()]
)  # reverse the index

# %%
data = pd.read_csv("disease_info.csv", encoding="cp1252")

# %%
from PIL import Image
import torchvision.transforms.functional as TF

# %%
def single_prediction(image_path):
    image = Image.open(image_path)
    image = image.resize((224, 224))
    input_data = TF.to_tensor(image)
    input_data = input_data.view((-1, 3, 224, 224))
    output = model(input_data)
    output = output.detach().numpy()
    index = np.argmax(output)
    print("Original : ", image_path[12:-4])
    pred_csv = data["disease_name"][index]
    print(pred_csv)
```

```
# %%
single_prediction("test_images/Apple_ceder_apple_rust.JPG")

# %% [markdown]
# ### Wrong Prediction

# %%
single_prediction("test_images/Apple_scab.JPG")

# %%
single_prediction("test_images/Grape_esca.JPG")

# %%
single_prediction("test_images/apple_black_rot.JPG")

# %%
single_prediction("test_images/apple_healthy.JPG")

# %%
single_prediction("test_images/background_without_leaves.jpg")

# %%
single_prediction("test_images/blueberry_healthy.JPG")

# %%
single_prediction("test_images/cherry_healthy.JPG")

# %%
single_prediction("test_images/cherry_powdery_mildew.JPG")

# %%
single_prediction("test_images/corn_cercospora_leaf.JPG")
```

```
# %%
single_prediction("test_images/corn_common_rust.JPG")

# %%
single_prediction("test_images/corn_healthy.jpg")

# %%
single_prediction("test_images/corn_northen_leaf_blight.JPG")

# %%
single_prediction("test_images/grape_black_rot.JPG")

# %%
single_prediction("test_images/grape_healthy.JPG")

# %%
single_prediction("test_images/grape_leaf_blight.JPG")

# %%
single_prediction("test_images/orange_haunglongbing.JPG")

# %%
single_prediction("test_images/peach_bacterial_spot.JPG")

# %%
single_prediction("test_images/peach_healthy.JPG")

# %%
single_prediction("test_images/pepper_bacterial_spot.JPG")

# %%
single_prediction("test_images/pepper_bell_healthy.JPG")
```

```
# %%
single_prediction("test_images/potato_early_blight.JPG")

# %%
single_prediction("test_images/potato_healthy.JPG")

# %%
single_prediction("test_images/potato_late_blight.JPG")

# %%
single_prediction("test_images/raspberry_healthy.JPG")

# %%
single_prediction("test_images/soyaben healthy.JPG")

# %%
single_prediction("test_images/potato_late_blight.JPG")

# %%
single_prediction("test_images/squash_powdery_mildew.JPG")

# %%
single_prediction("test_images/starwberry_healthy.JPG")

# %%
single_prediction("test_images/starwberry_leaf_scorch.JPG")

# %%
single_prediction("test_images/tomato_bacterial_spot.JPG")

# %%
single_prediction("test_images/tomato_early_blight.JPG")
```

```
# %%
single_prediction("test_images/tomato_healthy.JPG")

# %%
single_prediction("test_images/tomato_late_blight.JPG")

# %%
single_prediction("test_images/tomato_leaf_mold.JPG")

# %%
single_prediction("test_images/tomato_mosaic_virus.JPG")

# %%
single_prediction("test_images/tomato_septoria_leaf_spot.JPG")

# %%
single_prediction("test_images/tomato_spider_mites_two_spotted_spider_mites.JPG")

# %%
single_prediction("test_images/tomato_target_spot.JPG")

# %%
single_prediction("test_images/tomato_yellow_leaf_curl_virus.JPG")
```